View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VÝVOJ KALKULÁTORU PRO HODNOCENÍ ZRANITELNOSTÍ V JAVASCRIPTU

DEVELOPMENT OF A CALCULATOR FOR ASSESSING VULNERABILITIES IN JAVASCRIPT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Pavel Škrhák

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Eva Holasová

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Pavel Škrhák ***ID:*** 211583
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Vývoj kalkulátoru pro hodnocení zranitelností v Javascriptu

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace webové aplikace pro výpočet závažnosti zranitelnosti podle dnes standardně používaných metod (CVSS, OWASP Risk Rating Methodology atd.) v programovacím jazyce JavaScript. Grafické uživatelské rozhraní bude umožňovat zadání parametrů potřebných k výpočtu výsledného skóre a výsledky bude možné uložit do databáze. V teoretické části student provede analýzu současných webových zranitelností, nastuduje a analyzuje dnes používané metodologie k hodnocení zranitelností. Dále student analyzuje možnosti frameworku Vue JS. V praktické části student navrhne grafickou vizualizaci nástroje a následně implementuje za pomoci frameworku Vue JS funkční aplikaci kalkulátoru pro vypočet závažnosti zranitelností. Aplikace bude obsahovat nejméně dvě metody výpočtu. Aplikace bude také umožňovat získání výsledků uložených v databázi.

DOPORUČENÁ LITERATURA:

[1] HANCHETT, Erik a Benjamin LISTWON. Vue.js in Action. Manning Publications, 2019, 375 s. ISBN 9781617294624.

[2] WANG, Tingting, Qiujian LV, Bo HU a Degang SUN. CVSS-based Multi-Factor Dynamic Risk Assessment Model for Network System. 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC) [online]. IEEE, 2020, 289-294. DOI: 10.1109/ICEIEC49280.2020.9152340. ISBN 978--7281-6312-3.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Eva Holasová
Konzultant: Roman Kümmel (HACKER Consulting s.r.o.)

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:
Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

Cílem práce je popsat známé metody hodnocení zranitelností, a provést jejich implementaci do webové aplikace využívající framework Vue.js.

Práce popisuje dva systémy hodnocení zranitelností, a to CVSS (Common Vulnerability Scoring System) a OWASP (Open Web Application Security Project) Risk Rating Methodology. Jsou popsány jejich části, metriky a metody samotného výpočtu hodnocení. Následně jsou tyto systémy porovnány a jsou určeny jejich silné a slabé stránky. Práce dále hodnotí některé známe zranitelnosti pomocí těchto dvou metod hodnocení.

Práce dále popisuje návrh frontendu a backendu webové aplikace. Pro frontend je využit framework Vue.js, který umožňuje tvorbu dynamických jednostránkových webových aplikací. Jsou navrhnuty komponenty a rozložení aplikace. Dále je proveden návrh vzhledu frontednové aplikace a jejích komponentů. Backend byl navrhnut tak, aby vyhovoval frameworku Djnago, který spolu s frameworkem django REST framework slouží k rychlému vytváření API (Application Programming Interface) komunikujícího s databází. Byl navrhnut model pro ukládání dat z frontendové aplikace.

Následně práce popisuje samotnou implementaci této aplikace rozdělenou na frontend a backend. V backendu je popsána implementace API a databáze. Je popsána implementace samotného modelu, serializátoru a metod pro komunikaci s frontendovou aplikací. Ve frontendu je vytvořen vue router, který slouží k dynamické změně obsahu stránky, a poté již samotné komponenty, které slouží jako stavební bloky aplikace. Tyto komponenty obsahují tři části, a to strukturu, kód JavaScriptu a CSS (Cascading Sytle Sheets). Komponenty si mohou předávat data a volat funkce jiných komponentů.

Poslední částí práce je testování aplikace samotné. Je otestována její funkčnost pomocí výpočtu skóre již hodnocených zranitelností a některých bodů OWASP ASVS (Application Security Verification Standart). Dále je otestována bezpečnost pomocí testu několika známých zranitelností, společně s testováním pomocí OWASP ASVS.

Klíčová slova

CVSS, Django REST Framework, hodnocení, OWASP, Risk Rating Methodology, riziko, skóre, Vue.js, webová aplikace, zranitelnost

Abstract

The aim of this work is to describe the known methods of vulnerability assessment, and to implement them in a web application using the Vue.js framework.

The thesis describes two vulnerability assessment systems, namely CVSS (Common Vulnerability Scoring System) and OWASP (Open Web Application Security Project) Risk Rating Methodology. Their parts, metrics and methods of calculation of the evaluation are described. Subsequently, these systems are compared and their strengths and weaknesses are determined. The work then evaluates some known vulnerabilities using these two assessment methods.

The work then describes the design of the frontend and backend of the web application. The frontend uses the Vue.js framework, which allows the creation of dynamic one-page web applications. The components and layout of the application are designed. Furthermore, the appearance of the front application and its components is designed. The backend was designed to suit with the Djnago framework, which together with the django REST framework can be used to quickly create an API (Application Programming Interface) communicating with the database. A model for storing data from a frontend application was designed.

The work then describes the implementation of this application divided into frontend and backend. The backend describes the implementation of the API and the database. The implementation of the model itself, serializer and methods for communication with the frontend application are described. In the frontend, a vue router is created, which is used to dynamically change the content of the page, then the components themselves are created, which serve as building blocks of the application. These components contain three parts, namely structure, JavaScript code and CSS (Cascading Sytle Sheets). Components can pass data and call functions of other components.

The last part of the work is testing of the application itself. Its functionality is tested by calculating the score of already assessed vulnerabilities and some items of the OWASP ASVS (Application Security Verification Standard). Furthermore, security is tested by testing several known vulnerabilities, along with testing with OWASP ASVS.

Keywords

CVSS, Django REST framework, assessment, OWASP, Risk Rating Methodology, risk, score, Vue.js, Web application, vulnerability

Bibliografická citace

ŠKRHÁK, Pavel. *Vývoj kalkulátoru pro hodnocení zranitelností v Javascriptu.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/133535. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Eva Holasová.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Pavel Škrhák
VUT ID studenta:	211583
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Vývoj kalkulátoru pro hodnocení zranitelností v Javascriptu

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 24. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucí bakalářské práce Ing. Evě Holasové za cennou pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 24. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

Úvod

V této době, kdy se stále vyvíjí nové technologie a stále více věcí se přesouvá na internet, se potřeba zabezpečení stále zvedá. Vzniká velké množství webových aplikací, které mají stále více komplexních a důležitých úkolů, od jednoduchých webových kalkulaček, po komplikované informační systémy. Se stále větším množstvím důležitých dat přístupných z internetu, se však také zvedá množství zranitelností, kterých je možné využít pro získání těchto dat.

Dle Help Net Security bude konečné množství nově nalezených zranitelností v roce 2020 přes 20 000 [1]. Jelikož ale ne každá zranitelnost je stejně nebezpečná, bylo třeba vytvořit metody pro jejich hodnocení. Toto hodnocení určuje, jaký konečný dopad by mohl případný útok mít. Dvě nejznámější z těchto metod jsou popsány v této práci. Metoda Common Vulnerability Scoring System je nejpoužívanější metodou pro hodnocení, a stala se standardem pro mnoho databází zranitelností, včetně CVE (Common Vulnerabilities and Exposures) a NVD (National Vulnerability Database).

Tato práce má za cíl zanalyzovat používané metody hodnocení zranitelností, provést jejich srovnání a implementovat webovou aplikaci, která bude tyto dvě metody využívat. Dalším z cílů je také prozkoumání známých zranitelností, týkajících se webových aplikací a návrhu webové aplikace, která bude metody hodnocení implementovat.

V prvních dvou kapitolách je provedena analýza dvou metod pro hodnocení zranitleností, Jedná se o metody Common Vulnerability Scoring System a OWASP (Open Web Application Security Project) Risk Rating Methodology. Jsou popsány jejich části, faktory a metody výpočtu. Dále je proveden popis jejich výhod a nevýhod. Ve třetí kapitole je provedeno porovnání těchto dvou metod a srovnání některých jejich podobných faktorů. Je také provedeno porovnání možností jejich použití, které zahrnuje i jiné, méně známé metody hodnocení. Čtvrtá kapitola je zaměřená na popis známých zranitelností webových aplikací. Je uveden jejich popis a možnosti obrany proti nim. Pro všechny zranitelnosti je také provedeno hodnocení, pomocí výše uvedených metod. Pátá kapitola popisuje návrh webové aplikace. Je proveden návrh potřebných komponentů, jejich objektů a funkcí. Dále je proveden návrh grafického uživatelského rozhraní a backendu aplikace. V šesté kapitole je už popsána samotná implementace aplikace. Jako první je vysvětlena implementace backendu a databáze. Je vysvětleno vytvoření modelu, serializátoru a metod pro API (Application Programming Interface). Následuje implementace frontendu, vytvoření komponentů, příslušných metod a vzhledu. Poslední kapitola obsahuje testování samotné aplikace. Jsou vypočteny hodnoty pro zranitelnosti hodnocené v předchozích kapitolách a porovnány se správnými hodnotami. Dále jsou ohodnoceny některé body OWASP ASVS (Application Securtiy Verification Standard). Některé z těchto hodnot jsou uloženy do databáze, pro otestování její funkčnosti a funkčnosti API. Následuje testování bezpečnosti, kde jsou otestovány některé běžné zranitelnosti a vyhodnoceny některé požadavky OWASP ASVS.

1. COMMON VULNERABILITY SCORING SYSTEM

CVSS (Common Vulnerability Scoring System) je jednou z používaných metodik pro hodnocení závažnosti zranitelností [2]. CVSS přináší možnost, jak zachytit základní vlastnosti zranitelnosti a přetvořit je v numerické skóre o hodnotách od 0 do 10, které přímo odpovídá její závažnosti. Numerické skóre může být také přeloženo do kvalitativní reprezentace (qualitative representation), jako např.: nízká, střední a vysoká závažnost, aby pomohla organizacím správně posoudit a řídit jejich proces nakládání se zranitelnostmi.

Tento systém se zabývá třemi odlišnými oblastmi hodnocení. Těmi jsou základní skóre, dočasné skóre a skóre prostředí [4]. Každá z těchto oblastí spojuje související metriky, které zachycují charakteristiku zranitelnosti. Každá z těchto metrik má určité hodnoty, kterých může nabývat, a každá oblast má unikátní metodu výpočtu jejího skóre [4]. Skóre ze všech tří oblastí se poté kombinuje do výsledného skóre závažnosti zranitelnosti.

1.1 Verze Common Vulnerability Scoring System

CVSS se stále vyvíjí. Momentálně používaná je verze 3.1, verze 4 je ve vývoji [4]. Vývoj CVSS vychází ze zpětné vazby uživatelů, kteří tuto metodiku používají [3]. Většinou se změny týkají pouze metrik, případně přidání metriky do oblasti, ale objevují se i větší změny. První verze vznikla převážně kvůli sjednocení hodnocení rizik a tím urychlení jejich řešení a komunikaci mezi jednotlivými skupinami, které se zranitelnostmi zabývaly [5]. Druhá verze se zabývala změnou CVSS, aby mohlo být používáno ve více oblastech než původně [6], [7]. Bylo mnoho návrhů na různé změny a finálním výsledkem byly změna logiky za samotným výpočtem, změna některých metrik a tím zpřesnění hodnocení. Poslední velká změna byla mezi verzí 2.0 a 3.0 [7], [8]. Nejdůležitější z těchto změn jsou:

- přidání hodnoty fyzický (*Physical*) v metrice vektoru útoku (*Attack Vector*), tato hodnota určuje, že je třeba mít fyzický přístup ke zranitelnému komponentu,
- odebrání hodnoty střední (*Medium*) v metrice komplexita útoku (*Attack Complexity*),
- odstranění metriky autentizace (*Authentication*) a její nahrazení metrikou požadované oprávnění (*Privileges Required*), které hodnotí potřebné oprávnění pro útok,
- přidání metriky interakce uživatele (*User Interaction*), která hodnotí potřebnou interakci od uživatele, pro úspěšný útok,

- přidání metriky rozsah (*Scope*), která hodnotí dopad zranitelnosti i mimo daný zranitelný komponent systému (zranitelnost ve virtuálním systému zpřístupní i data na hostujícím systému),
- změna metriky exploitace (*Exploitability*) na kvalitu kódu exploitu (*Exploit Code Maturity*)
- odstranění metrik potenciální vedlejší škody (*Collateral Damage Potential*) a distribuce cíle (*Target Distribution*) a nahrazení modifikovanými metrikami základního skóre

1.2 Common Vulnerability Scoring System verze 3.1

CVSS verze 3.1 má lépe vysvětlené co která metrika obsahuje a je jednodušší pro pochopení, kterou hodnotu je třeba použít [3]. Byly provedeny menší změny ve výpočtu skóre, ale metriky a jejich hodnoty nebyly nijak změněny [4], [9].

Součástí CVSS je také CVSS vektor, což je řetězec znaků, který odráží názvy a hodnoty jednotlivých metrik, které byly využity pro výpočet závažnosti zranitelnosti [3]. Příkladem tohoto vektoru může být CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:L, kdy AV značí vektor útoku (Attack Vector) a L jeho hodnotu, tedy lokální (Local). Ostatní parametry toho řetězce se poté vyhodnotí sejným způsobem. V závislosti na množství zadaných informací, se poté mění délka tohoto řetězce [4].

1.2.1 Základní skóre

Základní skóre se zajímá o zranitelnost jako takovou a nepočítá s žádnými vnějšími vlivy [3], [4]. Základní skóre se v průběhu času a změně prostředí (například vlastní nastavení firewallu) nemění [4]. Základní skóre je možné stanovit okamžitě po objevení zranitelnosti.

Prvním ze tří hlavních prvků základního skóre je exploitovatelnost (*exploitability*) [4]. Ta se vztahuje přímo ke zranitelnému komponentu a její metriky jsou vektor útoku, komplexita útoku, vyžadované oprávnění a interakce uživatele. Druhým hlavním prvkem je rozsah, ten určuje, zda se zranitelnost může mít dopad i na jiný komponent, než jen zranitelný (zranitelnost virtuálního operačního systému zpřístupní data na hostitelském operačním systému) [4]. Rozsah samotný nemá žádné hodnocení, ale v závislosti na své hodnotě může změnit hodnocení ostatních metrik. Posledním prvkem je dopad. Ten určuje, jaký efekt bude mít úspěšné využití zranitelnosti na daný systém. Dopad se skládá ze tří metrik, a to dopad na důvěrnost, dopad na integritu a dopad na dostupnost. K výpočtu základního skóre je však nutné stanovit hodnoty jednotlivých metrik [4]. K tomu slouží tab. 1.1.

Tab. 1.1: Hodnocení základních metrik [3]

Název metriky	Popis	Možné hodnoty	Hodnocení
Vektor útoku	Přes jaké médium může být útok proveden	Vnější síťový	0,85
		Vnitřní síťový	0,62
		Lokální	0,55
		Fyzický	0,20
Komplexita útoku	Jak složité je uskutečnění útoku	Nízká	0,77
		Vysoká	0,44
Vyžadované oprávnění	Jaké oprávnění musí útočník mít, aby mohl provést útok	Žádné	0,85
		Nízké	0,62 (0,68 pokud je rozsah změněný)
		Vysoké	0,27 (0,5 pokud je rozsah změněný)
Interakce uživatele	Zda je vyžadována interakce uživatele	Žádná	0,85
		Vyžadovaná	0,62
Rozsah	Zda má zranitelnost dopad i na jiný, než zranitelný komponent	Změněný	
		Nezměněný	
Dopad na důvěrnost/ integritu/dostupnost	Jaký dopad bude útok mít na důvěrnost, integritu a dostupnost	Vysoký	0,56
		Nízký	0,22
		Žádný	0,00

Výpočet exploitovatelnosti je v celku jednoduchý a zahrnuje pouze vynásobení svých metrik. Pro exploitovatelnost bude zvolena zkratka E, poté VU pro vektor útoku, KU pro komplexitu útoku, PO pro požadované oprávnění a IU pro interakci uživatele. Rovnice pro výpočet má poté následující tvar [4].

$$E = 8{,}22 \cdot VU \cdot KU \cdot PO \cdot IU \tag{1.1}$$

Dalším krokem při výpočtu dopadu je výpočet sub skóre dopadu. Prvky tohoto skóre jsou hodnoty dopadu na důvěrnost, integritu a dostupnost. Toto skóre bude označeno zkratkou DSS, dále se označí dopad na důvěrnost jako DD, dopad na integritu jako DI a dopad na dostupnost jako DO. Sub skóre dopadu poté bude spočteno podle následující rovnice [4].

$$DSS = 1 - [(1 - DD) \cdot (1 - DI) \cdot (1 - DO)] \tag{1.2}$$

Samotný výpočet dopadu poté závisí na sub skóre dopadu a rozsahu. Pro dopad bude užita zkratka D a pro sub skóre dopadu zůstává zkratka DSS. Poté, pokud je rozsah nezměněný bude použita rovnice 1.3, zatímco při změněném rozsahu rovnice 1.4 [4].

$$D = 6{,}42 \cdot DSS \tag{1.3}$$

$$D = 7{,}52 \cdot (DSS - 0{,}029) - 3{,}25 \cdot (DSS - 0.02)^{15} \tag{1.4}$$

Pro konečný výpočet základního skóre jsou užity dvě funkce, a to *Roundup* a *Minimum*. Funkce *Roundup* zaokrouhlí číslo na 1 desetinné místo a zaokrouhluje jen k vyšším číslům. Funkce *Minimum* vrací menši ze dvou vstupních argumentů. Pro základní skóre bude zvolena zkratka ZS, pro funkci *Roundup* zkratka RndU, pro funkci *Minimum* zkratka Min, a pro exploitovatelnost a dopad budou ponechány zkratky E a D. Základní skóre poté opět závisí na hodnotě rozsahu, kdy bude užita rovnice 1.5, pokud je rozsah nezměněný a rovnice 1.6, pokud je rozsah změněný [4].

$$ZS = RndU(Min[(D + E), 10] \tag{1.5}$$

$$ZS = RndU(Min[1{,}08 \cdot (D + E), 10] \tag{1.6}$$

1.2.2 Dočasné skóre

Dočasné skóre už nehodnotí zranitelnost jako takovou, ale její závažnost s ohledem na vnější vlivy. Mezi tyto vlivy patří kvalita kódu exploitu, dostupnost záplat a množství informací o zranitelnosti, které jde v dané době získat [4]. Tyto vlivy mohou celkovou závažnost snížit (dostupnost záplat na danou zranitelnost), nebo zvýšit (dostupnost kvalitního exploitu) [3][4]. Pro výpočet dočasného skóre jsou využity hodnoty v tab. 1.2.

Tab. 1.2: Hodnocení dočasných metrik [3]

Název metriky	Popis	Možné hodnoty	Hodnocení
Kvalita kódu exploitu	Jaká je pravděpodobnost exploitu v závislosti na jeho kvalitě, dostupnosti kódu a jednoduchosti použití	Nedefinováno	1,00
		Bez důkazu	0,91
		Proof-of-Concept	0,94
		Funkční	0,97
		Vysoká kvalita	1,00
Úroveň nápravy	Vyjadřuje, v jakém stavu je momentální úroveň opravy zranitelnosti	Nedefinováno	1,00
		Nedostupné	1,00
		Dočasná záplata	0,96
		Workaround	0,97
		Oficiální záplata	0,95
Důvěrnost reportu	Říká, kolik a jak spolehlivých informací o exploitu je dostupných	Nedefinováno	1,00
		Neznámé	0,92
		Přiměřené	0,96
		Potvrzené	1,00

Výpočet dočasného skóre je v celku jednoduchý, v podstatě se jen vynásobí hodnoty jednotlivých metrik a základní skóre. Pro dočasné skóre bude zvolena zkratka DS, poté KE pro kvalitu kódu exploitu, UN pro úroveň nápravy, DR pro důvěrnost reportu a bude ponechána zkratka RndU pro funkci *Roundup*. Pro základní skóre bude použita zkratka DS. Výpočet je možné provést dle následující rovnice [4].

$$DS = RndU(ZS \cdot KE \cdot UN \cdot DR) \tag{1.7}$$

1.2.3 Skóre prostředí

Skóre prostředí závisí na prostředí, kde se zranitelnost nachází. Toto skóre se na rozdíl od dočasného skóre liší v různých výskytech dané zranitelnosti, protože každý systém může být jinak důležitý, nebo se můžou lišit bezpečnostní opatření systému [3]. U tohoto skóre je zvláště důležité, aby jej stanovila osoba, která má znalosti o daném systému a jeho zabezpečení. Skóre prostředí se z velké části skládá ze stejných parametrů jako základní skóre. Mění se ale v závislosti na prostředí, a se změnou prostředí se může změnit i základní skóre. Tyto nové hodnoty je možné poznat dle toho, že mají k názvu přidáno *Modified* (*Modified Base Metrics, Modified Attack Vector*) [4]. Hodnoty modifikovaného základního skóre se mohou lišit od základního skóre, ale jejích číselné hodnocení má stejnou váhu. Krom modifikovaného základního skóre má skóre prostředí ještě tři další parametry, které mají své hodnoty v tab. 1.3 [4].

Tab. 1.3: Hodnocení metrik prostředí [3]

Název metriky	Popis	Možné hodnoty	Hodnocení
Požadavek na důvěrnost/integritu/dostupnost	Jaká je požadovaná úroveň důvěrnosti, integrity či dostupnosti	Nedefinováno	1,00
		Nízký	0,50
		Střední	1,00
		Vysoký	1,50

Výpočet modifikované exploitovatelnosti je v podstatě stejný, jako předešlý výpočet. Je však třeba využít modifikované hodnoty [4]. Jelikož je množství prvků v tomto skóre velké, jsou zkraty uvedeny v tab. 1.4. K výpočtu je poté využita rovnice 1.8.

Tab. 1.4: Zkratky hodnot skóre prostředí

Hodnota	Zkratka
Sub skóre modifikovaného dopadu	MDSS
Požadavek na důvěrnost	PDD
Modifikovaný dopad na důvěrnost	MDD
Požadavek na integritu	PDI
Modifikovaný dopad na integritu	MDI
Požadavek na dostupnost	PDO
Modifikovaný dopad na dostupnost	MDO
Modifikovaný dopad	MD
Modifikovaná exploitovatelnost	ME

Modifikovaný vektor útoku	MVU
Modifikovaná komplexita útoku	MKU
Modifikované požadované oprávnění	MPO
Modifikovaná interakce uživatele	MIU
Kvalita kódu exploitu	KE
Úroveň nápravy	UN
Důvěrnost reportu	DR
Skóre prostředí	SP
Roundup	RndU
Minimum	Min

$$ME = 8{,}22 \cdot MVU \cdot MKU \cdot MPO \cdot MIU \tag{1.8}$$

Dalším krokem je výpočet sub skóre modifikovaného dopadu. Toto sub skóre vyjadřuje závislost požadavku a modifikovaného dopadu na důvěrnost, integritu a dostupnost [4]. Výpočet vyjadřuje následující rovnice.

$$MDSS = Min(\\ 1 - [(1 - PDD \cdot MDD) \cdot (1 - PDI \cdot MDI) \cdot (1 - PDO \cdot MDO)], 0{,}915) \tag{1.9}$$

Výpočet modifikovaného dopadu závisí na sub skóre modifikovaného dopadu a hodnotě modifikovaného rozsahu. Pokud má modifikovaný rozsah hodnotu nezměněný, bude použita rovnice 1.10, pokud má hodnotu změněný, je třeba použít rovnici 1.11 [4].

$$MD = 6{,}42 \cdot MDSS \tag{1.10}$$

$$MD = 7{,}52 \cdot (MDSS - 0{,}029) - 3.25 \cdot (MDSS \cdot 0.9731 - 0{,}02)^{13} \tag{1.11}$$

Posledním krokem ve výpočtu je pak už jen výpočet samotného skóre prostředí. Toto skóre také závisí na modifikovaném rozsahu. Pokud je rozsah nezměněný, bude využita rovnice 1.12, zatímco pro hodnotu změněný rovnice 1.13 [4].

$$SP = RndU(RndU[Min([MD + ME], 10)] \cdot KE \cdot UN \cdot DR) \tag{1.12}$$

$$SP = RndU(RndU[Min(1{,}08 \cdot [MD + ME], 10)] \cdot KE \cdot UN \cdot DR) \tag{1.13}$$

1.3 Common Vulnerability Scoring System verze 4.0

I když je stále používána verze 3.1 tohoto systému, verze 4.0 je již ve vývoji a jsou shromažďována vylepšení pro tuto novou verzi [4]. Tato verze by měla přinést nové možnosti, jak zranitelnost ohodnotit, a tím přinést změny do míry přesnosti hodnocení [10]. Mezi potvrzené změny patří například:

- výměna dočasných metrik za novou skupinu metrik hrozby (*Threat Metric Group*),
- odstranění metrik úroveň nápravy (*Remedation Level*) a důvěrnost reportu (*Report Confidence*),
- výměna metriky kvalita kódu exploitu (*Exploit Code Maturity*) za *Exploit Maturity*, neboli kvalita exploitu samotného,
- změna hodnot rozsahu (*Scope*) na tři hodnoty.

Krom potvrzených změn obsahuje dokument vylepšení i mnoho potencionálních změn, které musí být přijaty, či odmítnuty [10]. Verze 4.0 by tedy mohla potencionálně přinést tomuto sytému mnoho výhod a odstranit některé jeho nevýhody, a tím ještě zvýšit míru použití tohoto systému.

1.4 Výhody a nevýhody Common Vulnerability Scoring System

Hlavní výhodou metodiky CVSS je její komplexita a fakt, že se stále vyvíjí. V případě použití této metodiky zkušenou a znalou osobou, která správně použije všechny metriky, je možné získat velmi přesné skóre závažnosti dané zranitelnosti [3]. Pokud však tuto metodiku použije nezkušená osoba, která nemá veškeré znalosti o systému a prostředí, lze získat skóre, které závažnosti zranitelnosti vůbec neodpovídá. Hlavní nevýhodou této metodiky je fakt, že je již vcelku zastaralá [11]. Tento problém by však měla vyřešit verze 4.0. Další nevýhodou této metody je, že není určena k zjištění priority zranitelnosti, ale pouze k posouzení její závažnosti [11]. Jednou z často zmiňovaných stížností je také fakt, že skóre ne vždy odpovídá pravé závažnosti. Dle výzkumu Tenable, je až 56 % všech nových zranitelností ohodnoceno s výsledným skóre 7–10, i když tyto zranitelnosti nemusí být nikdy reálně zneužity, což také souvisí s faktem, že k 75 % těchto zranitelností nebyl nikdy zveřejněn exploit [11]. Jednou z dalších nevýhod je také fakt, že skóre se často vypočítá pouze jednou a poté se již nemění, což může vést ke zranitelnostem s vyšším, či nižším skóre, než by měly mít [11].

2. OPEN WEB APPLICATION SECURITY PROJECT RISK RATING METHODOLOGY

OWASP Risk Rating Methodology je jednou z dalších možností, jak hodnotit riziko zranitelností. Tato metodologie se zaměřuje primárně na zranitelnost webových aplikací, ale dá se použít i pro hodnocení jiných zranitelností. I když tato metodologie není tak rozšířená jako CVSS, poskytuje i jiný pohled na riziko zranitelnosti než CVSS [3]. Základem této metodologie je vztah mezi rizikem, pravděpodobností a dopadem zneužití zranitelnosti, což jsou také dvě hlavní části výpočtu [3], [12]. Prvním krokem před samotným výpočtem je však zjišťování informací o riziku, které je potřeba hodnotit. Osoba, která bude provádět test by měla zjistit informace o potenciálních útočnících, útoku, který může útočník provést, a zranitelnosti, na kterou může být útok proveden. Dalším krokem je odhad dopadu, který může úspěšný útok mít [12].

2.1 Pravděpodobnost zneužití zranitelnosti

Pokud osoba, která hodnocení provádí, má dostupné informace o možném útočníkovi a zranitelnosti, může začít s hodnocením pravděpodobnosti zneužití zranitelnosti. Pravděpodobnost zneužití zranitelnosti, je v podstatě odhad toho, jak pravděpodobné je nalezení a zneužití zranitelnosti útočníkem. Tato hodnota se skládá ze dvou hlavních prvků, to jsou faktory nositele hrozby a faktory zranitelnosti. Z hodnot těchto prvků je poté vypočten průměr, který je použit k vlastnímu hodnocení [12].

Faktory nositele hrozby

Tyto faktory se vztahují k nositeli hrozby (útočníkovi), a jejich cíl je ohodnotit pravděpodobnost úspěšného útoku na základě schopností a vlastností útočníka [12]. Jejich hodnoty jsou uvedeny v tab. 2.1.

Tab. 2.1: Faktory nositele hrozby [3]

Faktor	Popis	Možné hodnoty	Hodnocení
Úroveň schopností	Úroveň schopností nositele hrozby	Žádné technické schopnosti	1,00
		Nízké technické schopnosti	3,00
		Pokročilý uživatel počítačových zařízení	5,00
		Znalost sítí a programování	6,00

		Znalost penetračního testování	9,00
Motivace	Jak motivovaný je nositel hrozby	Nízký, nebo žádný zisk	1,00
		Průměrný zisk	4,00
		Vysoký zisk	9,00
Příležitost	Jaké zdroje a příležitosti jsou potřeba pro úspěšný útok	Je vyžadován plný přístup, nebo drahé zdroje	0,00
		Je vyžadován speciální přístup, nebo zdroje	4,00
		Je vyžadován jakýkoliv přístup, nebo zdroje	7,00
		Není vyžadován žádný přístup, ani zdroje	9,00
Skupina nositelů hrozby	Jaké vlastnosti má skupina nositelů hrozby	Vývojáři	2,00
		Systémoví správci	2,00
		Uživatelé intranetu	4,00
		Partneři	5,00
		Autentizovaní uživatelé	6,00
		Anonymní uživatelé internetu	9,00

Faktory zranitelnosti

Tyto faktory se vztahují ke zranitelnosti samotné a jejím vlastnostem. Cíl je zhodnotit pravděpodobnost nalezení a zneužití této dané zranitelnosti, zatímco je brán v potaz agent, nebo agenti hrozby, kteří byli vybráni u předchozí sady faktorů [12]. Hodnoty jsou uvedeny v tab. 2.2.

Tab. 2.2: Faktory zranitelnosti [3]

Faktor	Popis	Možné hodnoty	Hodnocení
Jednoduchost odhalení	Jak jednoduché je odhalit zranitelnost pro nositele hrozby	V podstatě nemožné	1,00
		Těžké	3,00
		Jednoduché	7,00
		Jsou k dispozici automatické nástroje	9,00
Jednoduchost exploitace	Jak jednoduché je využít zranitelnosti	Teoretické	1,00
		Těžké	3,00
		Jednoduché	5,00
		Automatické nástroje k dispozici	9,00
Povědomí	Jak známá je zranitelnost pro nositele hrozby	Neznámá	1,00
		Skrytá	4,00
		Známá	6,00
		Veřejně známá	9,00
Detekce průniku	Jak pravděpodobné je detekování průniku	Aktivní detekce v aplikaci	1,00
		Logováno a prozkoumáno	3,00
		Logováno bez prozkoumání	8,00
		Nelogováno	9,00

2.2 Dopad zneužití zranitelnosti

Při hodnocení dopadu, je třeba brát v potaz fakt, že jsou přítomny dva typy dopadu. Prvním z nich je technický dopad, na samotnou aplikaci, její data a funkce které poskytuje. Druhým typem je pak dopad na byznys, který říká, jaký dopad má zranitelnost na byznys a společnost samotnou [12]. Dopad na byznys je více důležitý, ale vyžaduje velké množství informací, které nemusí být k dispozici. Proto, pokud je to možné, je využit dopad na byznys, jinak je použit technický dopad, který nevyžaduje takové množství informací. K vlastnímu hodnocení se také používá průměr hodnot, jako u pravděpodobnosti zneužití, v tomto případě ale jen z jednoho typu dopadu [12].

Faktory technického dopadu

Technický dopad může být rozdělen na faktory, které se tradičně v oblasti zabezpečení používají. Tyto faktory jsou důvěrnost, integrita, dostupnost a zodpovědnost [12]. Hodnoty jednotlivých faktorů jsou uvedeny v tab. 2.3.

Tab. 2.3: Faktory technického dopadu [3]

Faktor	Popis	Možné hodnoty	Hodnocení
Ztráta důvěrnosti	Kolik dat by mohlo být odhaleno a jak jsou citlivé	Odhaleno minimum necitlivých dat	2,00
		Odhaleno minimum citlivých dat	6,00
		Odhalena většina necitlivých dat	6,00
		Odhalena většina citlivých dat	7,00
		Odhalena všechna data	9,00
Ztráta integrity	Kolik dat může být poškozeno, a jak velké je poškození	Minimální množství málo narušených dat	1,00
		Minimální množství velmi narušených dat	3,00
		Velké množství málo narušených dat	5,00
		Velké množství vážně narušených dat	7,00
		Všechna data kompletně znehodnocena	9,00
Ztráta dostupnosti	Kolik služeb může být ztraceno a jak důležité jsou	Minimální přerušení sekundárních služeb	1,00
		Minimální přerušení primárních služeb	5,00
		Rozsáhlé přerušení sekundárních služeb	5,00
		Rozsáhlé přerušení primárních služeb	7,00
		Všechny služby ztraceny	9,00
Ztráta zodpovědnosti	Možnost vysledování nositele hrozby k jedinci	Dá se vysledovat	1,00
		Pravděpodobné vysledování	7,00
		Anonymní	9,00

Faktory dopadu na byznys

I když dopad na byznys vyplývá z technického dopadu, k jeho zhodnocení je třeba porozumění toho, co je důležité pro samotnou společnost provozující aplikaci. Hlavním cílem tohoto dopadu není zhodnotit technické riziko, ale možnost ztráty příjmů, zákazníků a reputace, které mají pro společnost mnohem větší význam [12]. Jednotlivé hodnoty těchto faktorů jsou uvedeny v tab. 2.4.

Tab. 2.4: Faktory dopadu na byznys [3]

Faktor	Popis	Možné hodnoty	Hodnocení
Finanční škody	Jak velké finanční škody mohou vzniknout ze zneužití zranitelnosti	Méně než náklady opravy zranitelnosti	1,00
		Malý efekt na roční zisk	3,00
		Značný efekt na roční zisk	7,00
		Bankrot	9,00
Škody na reputaci	Možné škody na reputaci ovlivňující zisk společnosti	Minimální škody	1,00
		Ztráta významných účtů	4,00
		Ztráta dobré pověsti	5,00
		Poškození značky	9,00
Nesoulad	Jak vážné by bylo nevyhovění požadavkům	Málo závažné porušení požadavků	2,00
		Zřetelné porušení požadavků	5,00
		Velmi závažné porušení požadavků	7,00
Narušení soukromí	Kolik soukromých informací by mohlo být odhaleno	Jedince	3,00
		Stovky osob	5,00
		Tisíců osob	7,00
		Milionů osob	9,00

2.3 Stanovení závažnosti rizika

Ke stanovení rizika je třeba spojit pravděpodobnost a dopad zneužití zranitelnosti. Nejdříve je ale třeba převést hodnoty těchto dvou oblastí a zjistit, zda je hodnota pravděpodobnosti a dopadu nízká, střední nebo vysoká [12]. Toho lze docílit pomocí tab. 2.5, která tyto hodnoty v rozsahu 0 až 9 rozděluje do tří částí.

Tab. 2.5: Zjištění hodnoty pravděpodobnosti a dopadu [12]

Úrovně pravděpodobnosti a dopadu	
0 až < 3	Nízká
3 až < 6	Střední
6 až 9	Vysoká

Pokud jsou známy hodnocení pravděpodobnosti a dopadu (ideálně dopad na byznys) zneužití zranitelnosti, je možné přejít k samotnému hodnocení závažnosti. To se provádí dle tab. 2.6.

Tab. 2.6: Určení závažnosti rizika dle metodologie OWASP [12]

Celková závažnost rizika				
Dopad	Vysoký	Střední	Vysoká	Kritická
	Střední	Nízká	Střední	Vysoká
	Nízký	Žádná	Nízká	Střední
		Nízká	Střední	Vysoká
	Pravděpodobnost			

Po určení závažnosti rizika, by se měla společnost rozhodnout, které z těchto rizik má nejvyšší prioritu na jeho odstranění, a zda se jeho odstranění vyplatí. Každá společnost by také ideálně měla mít přizpůsobený způsob hodnocení rizik [12].

2.4 Výhody a nevýhody Open Web Application Security Project Risk Rating Methodology

Hlavní výhodou této metodologie je hlavně její jednoduchost. Další výhodou je použití dopadu na byznys, díky kterému je výpočet závažnosti mnohem přesnější než jen s technickým dopadem, a je zde tudíž možnost tuto metodiku přizpůsobit potřebám jednotlivých organizací [3]. Velkou výhodou je také hodnocení samotného potencionálního útočníka. Výpočet této metodologie je také vcelku jednodušší než více rozšířené CVSS, a tím snadnější na použití [12]. Nelze také opomenout fakt, že OWASP Risk Rating Methodology se zabývá rizikem zranitelnosti, je tedy možné ji použít i k určení priority zranitelnosti [12]. I když je jednoduchost jednou z výhod této metodologie, je také jednou z jejích hlavních nevýhod, jelikož nemusí obsahovat všechny faktory potřebné pro zhodnocení [3]. Faktory, které jsou v CVSS hodnoceny několika metrikami, a můžeme tedy docílit vyšší přesnosti u konečného skóre, jsou v této metodice někdy opomenuty, a jindy zase shrnuty do jedné hodnoty [12].

3. POROVNÁNÍ METODIK PRO HODNOCENÍ ZRANITELNOSTÍ

Jelikož práce popisuje dvě hlavní metodiky hodnocení zranitelnosti, je vhodné provést jejich srovnání. Zatímco metodika CVSS je jedna z nejpoužívanějších metod pro hodnocení, nabízí OWASP Risk Rating Methodology pohled na zranitelnost i z jiného pohledu než CVSS [4], [12].

3.1 Rozdíly

Jedním z rozdílu je, že CVSS se využívá pro výpočet závažnosti zranitelnosti, zatímco OWAPS Risk Rating Methodology je využíván pro výpočet závažnosti rizika. Dalším z rozdílů je fakt, že metodika CVSS je více zaměřená na technickou stránku zranitelnosti a na samotný systém, kde se zranitelnost nachází [4]. S tím souvisí i faktory, na které se metodika zaměřuje. Zároveň má většina jejich metrik většinou dvě, nebo tři možné hodnoty. Základní skóre se zaměřuje na zranitelnost a co je třeba k jejímu zneužití. Dočasné skóre se zabývá opravou a dostupností kódu pro zranitelnost, zatímco skóre prostředí je zaměřené na vlastnosti systému [4]. Rozdíl je v konečném hodnocení, kdy výstupem CVSS je číselné hodnocení závažnosti společně s CVSS vektorem, zatímco výstupem OWASP Risk Rating Methodology je slovní hodnocení rizika, které z číselného vychází [2], [12].

OWASP Risk Rating Methodology je zaměřena jak na technickou stránku, tak i na případného útočníka a dopad na společnost, nebo člověka provozujícího zranitelný systém. Většina faktorů má také větší množství možných hodnot. Faktory nositele hrozby jsou zaměřeny na útočníka, či skupinu útočníků, jejich motivaci a schopnosti. Faktory zranitelnosti jsou zaměřeny na zranitelnost samotnou, její dostupnost a jednoduchost jejího zjištění a zneužití. Faktory technického dopadu jsou také zaměřeny na zranitelnost a technický dopad, podobně jako CVSS, měly by se však použít jen v případě, že nelze spolehlivě určit dopad na byznys [12]. Dopad na byznys je jednou z hlavních výhod OWASP Risk Rating Methodology. Pohlíží na zranitelnost nejen z technické stránky, ale také z pohledu na reputaci a finanční situaci společnosti, a tím jí dává možnost upravit priority dle rizika zranitelnosti [12].

3.2 Podobné vlastnosti

I přes rozdíly mají obě metody hodnocení podobné vlastnosti. Mezi podobné faktory pro obě metody patří faktory dopadu a komplexita útoku. OWAPS Risk Rating Methodology však nabízí větší rozsah hodnot své faktory [12]. Faktory a možné hodnoty jsou pro porovnání uvedeny v tab. 4.1. K porovnání byla uvedena jen komplexita útoku a dopad na důvěrnost, jelikož faktory dopadu mají stejné, nebo velmi podobné hodnocení.

Tab. 3.1: Porovnání podobných faktorů metodik

Faktory CVSS		Faktory OWASP Risk Rating Methodology	
Komplexita útoku	Nízká	Jednoduchost exploitace	Teoretické
			Těžké
	vysoká		Jednoduché
			Automatické nástroje k dispozici
Dopad na důvěrnost	Žádný	Ztráta důvěrnosti	Odhaleno minimum necitlivých dat
	Nízký		Odhaleno minimum citlivých dat
	Vysoký		Odhalena většina necitlivých dat
			Odhalena většina citlivých dat
			Odhalena všechna data

Jednou z dalších možností hodnocení zranitelností je také VRT (Vulnerability Rating Taxonomy). Tato metoda rozděluje zranitelnosti do pěti stupňů dle priority, od nejvyšší P1 (kritická), po nejnižší, P5 (akceptovaný risk) [13]. Podle této priority by měly být zranitelnosti odstraněny. Tato metoda je založena na principu "lovců odměn" (bounty hunters), kdy lidé mohou předkládat své nálezy zranitelností a získat za ně odměnu. Tato metoda používá k výslednému hodnocení priority mnoho dalších faktorů včetně CVSS skóre a dopadu. Tato metoda tedy není zaměřená přímo na hodnocení zranitelnosti, ale její priority. Jednou z hlavních částí této metody je komunikace, kdy hodnotitel musí komunikovat se zákazníkem, aby mohl správně zhodnotit prioritu. Zranitelnost by také měla být hodnocena dle prostředí, aby nedocházelo k podhodnocení některých zranitelností [14]. Tato metoda se stále vyvíjí a má mnoho verzí, poslední aktualizovanou v roce 2020. Dalším přídavkem bylo zahrnutí OWASP Top Ten, Top Ten Mobile a možnost zahrnutí dat z CWE (Common Weakness Enumeration). Organizace Bugcrowd také pravidelně pořádá konference, na kterých se posuzují navržené změny hodnocení a nově nalezené zranitelnosti, díky kterým je poté případně hodnocení upraveno [13].

Jsou tedy porovnány tři metody hodnocení zranitelností. Každá z metod dovoluje se zaměřit na jednu část. Na zranitelnost samotnou a její závažnost, poté na závažnost rizika zranitelnosti, a jako poslední na prioritu zranitelnosti. Tyto tři metody by tedy byly ideální pro kompletní hodnocení zranitelnosti.

4. WEBOVÉ ZRANITELNOSTI

Jako první je vhodné říct, co je zranitelnost. Zranitelnost je slabina, kterou může útočník využít k získání přístupu do systému, nebo provedení nějaké nepovolené akce. Existuje mnoho druhů zranitelností, které jsou charakteristické pro různé systémy, či služby. Tato práce se zabývá převážně webovými zranitelnostmi, tedy zranitelnostmi, které jsou zvláště nebezpečné pro webové stránky a webové aplikace. Kvůli rozdílu dopadu, který může zneužití zranitelnosti mít, se začalo hodnotit riziko, či závažnost zranitelnosti. Dvě různé metody hodnocení byly popsány v předchozích kapitolách.

4.1 SQL Injekce

Tato zranitelnost je jedna z nejznámějších a zároveň jedna z nejvíce se vyskytujících webových zranitelností. Tato zranitelnost je zaměřená na databáze napsané jazykem SQL (Structured Query Language). Princip spočívá ve vložení příkazu na vstup (vyhledávání v databázi). Pokud je útok úspěšný, databáze převezme příkaz, ale místo vyhledání tohoto řetězce jej provede jako příkaz. Tímto způsobem je možné vypisovat, přepisovat či mazat data z databáze. Je zde také možnost znepřístupnění databáze, nebo ovlivnění samotného systému, na kterém databáze běží [15].

Nejjednodušší ochranou proti tomuto útoku je použití tzv. *prepared statements*. U této metody je třeba nejdříve mít napsaný samotný SQL kód a jednotlivé parametry předat do dotazu až později. Případný kód, který je zadán, je tedy brán jako jeden řetězec, a ne jako kód. Jednou z dalších možností je použití uložených procedur. Tato metoda, pokud je naprogramovaná správně, má stejný efekt jako *prepared statements*. Je však třeba mít na paměti, že tato metoda může v některých případech riziko i zvýšit [16].

Hodnocení

Pro hodnocení je třeba nejdříve získat informace. Útok může být proveden odkudkoliv, dokud je přístup k databázi z internetu. Komplexita útoku je v celku nízká, příkazy se také dají jednoduše najít. K útoku nejsou v podstatě potřebná žádná práva, to se ale může změnit dle umístění databáze. V tomto případě bude dáno, že přístupová práva nejsou zapotřebí. Pro tento útok není interakce jiného uživatele zapotřebí, ale v některých případech může rozsah této zranitelnosti zasahovat i mimo samotnou databázi [17]. Útok má také vysoký dopad na důvěrnost, integritu a dostupnost dat. Díky těmto informacím je možné získat skóre pomocí CVSS (CVSS verze 3.1). Při použití této metody hodnocení je získáno skóre 10,0, tedy kritická závažnost zranitelnosti s CVSS vektorem CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H. Toto skóre se samozřejmě mění v závislosti na umístění této databáze v systému. Pokud je databáze dostupná například pouze ze sítě, na které databáze je a jsou potřebná práva k přístupu, je riziko pouze 6,5, tedy střední míra závažnosti. Hodnoty jsou také uvedeny v tab. 4.1.

Tab. 4.1: Hodnocení SQL Injekce dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	N
Rozsah	Změněný	C
Dopad na důvěrnost	Vysoký	H
Dopad na integritu	Vysoký	H
Dopad na dostupnost	Vysoký	H

U OWASP Risk Rating Methodology je situace složitější, jelikož je zde více vlastností souvisejících se samotným prostředím, kde se zranitelný systém nachází (například motiv, kdy je jednou z hlavních věcí „odměna“ kterou útočník získá). Z tohoto důvodu nebudou faktory nositele hrozby použity. Jelikož je útok známý, a existují i automatické nástroje pro útok, bude výsledné hodnocení pravděpodobnosti vysoké. Detekce útoku na výsledné skóre v tomto případě nemá vliv. Pro hodnocení dopadu bude použito technického dopadu. Jelikož je zde v celku velký dopad na důvěrnost, integritu a dostupnost a útočník může být anonymní, je výsledné hodnocení vysoké [17]. Díky hodnotám pravděpodobnosti a dopadu je dosaženo výsledné závažnosti zranitelnosti, která je v tomto případě kritická. Toto skóre se může samozřejmě změnit, při použití faktorů nositele hrozby. Hodnoty pro faktory zranitelnosti jsou uvedeny v tab. 4.2, zatímco hodnoty technického dopadu v tab. 4.3.

Tab. 4.2: Hodnoty faktorů zranitelnosti pro SQL Injekci

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Automatické nástroje k dispozici	9,00
Jednoduchost exploitace	Automatické nástroje k dispozici	9,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Aktivní detekce v aplikaci	1,00

Tab. 4.3: Hodnoty faktorů technického dopadu pro SQL Injekci

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhalena všechna data	9,00
Ztráta integrity	Všechna data kompletně znehodnocena	9,00
Ztráta dostupnosti	Všechny služby ztraceny	9,00
Ztráta zodpovědnosti	Anonymní	9,00

4.2 Cross-site scripting

Zranitelnost XSS (Cross-site Scripting) znamená, že webová aplikace, či stránka je zranitelná vůči injekci skriptu (JavaScript). Dá se tedy říct, že XSS je jednou z forem injekce. Útočník je schopný poslat škodlivý skript uživateli. Jeho prohlížeč se však nerozeznává, zda se jedná o legitimní skript, nebo ne. Prohlížeč poté tento skript provede [18].

Existuje několik typů XSS. Jeden typ je odražený (reflected) XSS (označováno jako typ 1), kdy webová aplikace obsahuje nekontrolovaný uživatelský vstup jako část HTML (Hypertext Markup Lanuage) a server tento vstup navrací. Úspěšný útok dovolí útočníkovi spustit libovolný skript v HTML nebo JavaScriptu. Dalším typem je uložený (stored) XSS (někdy nazýván typ 2). Tento typ je podobný předchozímu, není ale zobrazen pouze jedné oběti, ale je místo toho uložen na serveru aplikace, kde může být zobrazen všemi uživateli (blog s komentáři, kde skript je uložen v jednom z komentářů). Tento typ je nejnebezpečnější ze všech typů XSS [19][19]. Poslední typ je DOM (Document Object Model) XSS (někdy nazýván typ 0 [20]. Skript je proveden jako výsledek změny DOM prostředí v prohlížeči oběti (například URL).

Nejjednodušší metodou prevence této zranitelnosti je nevkládat nedůvěryhodná data do aktivních prvků HTML. Další metodou může být filtrace vstupních dat, nebo použití některé z dostupných bezpečnostních knihoven [21].

Hodnocení

Všechny tři typy mají několik společných vlastností. Všechny typy je možné zneužít přes síť. V této době je komplexita útoku v celku nízká a existují i automatické nástroje pro zjištění a realizaci útoku. Případný útočník také nepotřebuje žádná práva a zranitelnost má dopad i na jiné části systému. U typů 0 a 1 je třeba interakce ze strany uživatele, zatímco u typu 2 není, jelikož skript může být uložen třeba v komentáři a spustí se bez potřeby interakce uživatele. Žádný z těchto typů nemá žádný vliv na dostupnost, ale zatímco odražený a DOM XSS mají nízký dopad jak na důvěrnost, tak na integritu, má uložený XSS vysoký dopad na důvěrnost a žádný dopad na integritu. Pokud jsou tyto informace použity pro výpočet CVSS skóre, vychází skóre 6,1 pro odražený a DOM XSS a 8,6 pro uložený XSS. Pro odražený XSS jsou hodnoty pro hodnocení CVSS uvedeny v tab. 4.4, a výsledný CVSS vektor má následující tvar CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N [17].

Tab. 4.4: Hodnocení odraženého XSS dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	R
Rozsah	Nezměněný	C
Dopad na důvěrnost	Vysoký	L
Dopad na integritu	Vysoký	L
Dopad na dostupnost	Vysoký	N

Existence automatických nástrojů pro detekci a zneužití této zranitelnosti v kombinaci s celkovým povědomím o této zranitelnosti značí, že pravděpodobnost útoku na tuto zranitelnost je vysoká. I když má uložený XSS vysoký dopad na důvěrnost, nemá žádný dopad na integritu [17]. Odražený a DOM XSS mají nízký dopad na důvěrnost a integritu. Díky těmto rozdílům je výsledný dopad pro odražený a DOM XSS střední, zatímco pro uložený XSS vysoký. Díky těmto informacím je možné určit, že výsledná závažnost zranitelnosti je dle OWASP Risk Rating Methodology vysoká pro odražený a DOM XSS, a kritická pro uložený XSS. Hodnoty pro odražený XSS jsou uvedeny v tab. 4.5 a tab. 4.6.

Tab. 4.5: Hodnoty faktorů zranitelnosti pro odražený XSS

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Automatické nástroje k dispozici	9,00
Jednoduchost exploitace	Automatické nástroje k dispozici	9,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Aktivní detekce v aplikaci	1,00

Tab. 4.6: Hodnoty faktorů technického dopadu pro odražený XSS

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhaleno minimum citlivých dat	6,00
Ztráta integrity	Velké množství málo poškozených dat	5,00
Ztráta dostupnosti	Minimální ztráta sekundárních služeb	1,00
Ztráta zodpovědnosti	Anonymní	9,00

4.3 Cross Site Request Forgery

I když je tato zranitelnost a útok na ni v celku známý, je stále přítomna v mnoha webových stránkách a aplikacích. Tato zranitelnost umožňuje útočníkovi provést akci na webu, kde je oběť autorizovaná, s právy oběti[22]. Úspěšný útok na tuto zranitelnost má katastrofální dopad na důvěrnost a integritu, a v některých případech i na dostupnost (útok na web, kdy je poslán požadavek s autorizací administrátora). Útok na tuto zranitelnost vyžaduje jistou formu sociálního inženýrství. Prohlížeč u většiny webových stránek či aplikací automaticky odesílá i informace o uživateli (cookies relace atd.). Pokud jsou tyto informace zneužity k vytvoření jiného požadavku, webová stránka či aplikace většinou nemá žádnou možnost, jak jej rozlišit od legitimního požadavku od oběti. Útočník se u této zranitelnosti většinou snaží provést nějakou změnu. Tato změna může být cokoliv od změny uživatelského jména, až po nákup ve jménu oběti [22].

Jako ochrana proti této zranitelnosti nefunguje použití tajných cookie souborů (secret cookie), jelikož je prohlížeč stále odesílá při požadavku [22]. Jednou z možností ochrany proti této zranitelnosti je použití tokenů buď pro relaci, nebo pro každý požadavek [23]. I když je generace tokenu pro každý požadavek je bezpečnější, jelikož útočník nemá velké množství času, aby byl schopný tento token získat, je více využívaná generace tokenu pouze pro relaci. Token by neměl být přenášen pomocí cookies. Server webové stránky nebo aplikace kontroluje, zda se token shoduje s vygenerovaným. Pokud se token neshoduje, měl by být požadavek přerušen, relace ukončena a incident zalogován jako potencionální útok [23].

Hodnocení

Tato zranitelnost může být využita odkudkoliv, a i přes potřebu sociálního inženýrství, je komplexita útoku v celku nízká. Útok na tuto zranitelnost nevyžaduje žádná práva a rozsah dopadu této zranitelnosti se vztahuje pouze na zranitelnou stránku. Vyžaduje však interakci uživatele (kliknutí na odkaz). Největší dopad má tato zranitelnost na integritu, jelikož útočník může přímo modifikovat účet oběti na zranitelné stránce. I když útočník nemůže přímo získat důvěrná data (zobrazila by se oběti), je dopad na důvěrnost stále vysoký, jelikož útočník může data odtajnit či přeposlat. Dopad na dostupnost je také vysoký, útočník může přímo data mazat, nebo může provést příkaz s právy oběti. S těmito informacemi je možné určit skóre CVSS, které má výslednou hodnotu 8,8, s vektorem v následujícím tvaru CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H [24]. Hodnoty použité pro výpočet CVSS skóre jsou uvedeny v tab. 4.7.

Tab. 4.7: Hodnocení Cross Site Request Forgery dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	R
Rozsah	Nezměněný	U
Dopad na důvěrnost	Vysoký	H
Dopad na integritu	Vysoký	H
Dopad na dostupnost	Vysoký	H

Pro tuto zranitelnost jsou dostupné nástroje pro její detekci a provedení útoku. Tato zranitelnost je také poměrně známá, jsou ale nástroje určené pro detekci útoků na tuto zranitelnost. Jelikož je případný útočník anonymní, je výsledná závažnost dle OWASP Risk Rating Methodology kritická [24]. Tato hodnota se však může změnit při použití faktorů nositele hrozby. Hodnoty pro hodnocení pomocí OWASP Risk Rating Methodology jsou uvedeny v tabulkách 4.8 a 4.9.

Tab. 4.8: Hodnoty faktorů zranitelnosti pro Cross Site Request Forgery

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Automatické nástroje k dispozici	9,00
Jednoduchost exploitace	Automatické nástroje k dispozici	9,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Aktivní detekce v aplikaci	1,00

Tab. 4.9: Hodnoty faktorů technického dopadu pro Cross Site Request Forgery

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhalena všechna data	9,00
Ztráta integrity	Všechna data kompletně znehodnocena	9,00
Ztráta dostupnosti	Všechny služby ztraceny	9,00
Ztráta zodpovědnosti	Anonymní	9,00

4.4 CRLF Injekce

Zkratka CRLF značí Carriage Return a Line Feed, což jsou znaky, které určují ukončení řádku. Carriage Return (%0d, \r) znamená návrat na začátek řádku, a Line Feed (%0a, \n) značí posun na nový řádek [25]. Různé systémy mají jiné požadavky na ukončení řádku. HTTP (Hypertext Transfer Protocol) protokol sekvenci Carriage Return, Line Feed vždy považuje ze ukončení řádku. Útočník se tedy může pokusit poslat CRLF do aplikace, nebo webu [25]. Jedním z možných útoků je HTTP Response Splitting. Jelikož HTTP používá CRLF pro rozdělení hlavičky a těla, je útočník pomocí vložení CRLF schopný zmást HTTP protokol a vložit vlastní kód, který bude protokol brát jako tělo [26]. Dalším z možných útoků je Log Injection, díky kterému je útočník schopný vytvářet falešné záznamy v logu [27]. Obranou proti této zranitelnosti může být i pouhé ošetření vstupu proti těmto znakům [25].

Hodnocení

Útok na tuto zranitelnost je možné provést přes síť. Komplexita je přitom nízká a nejsou vyžadována žádná práva. Ostatní parametry se poté liší dle dalších faktorů. Pro příklad hodnocení bude použito CVE-2019-19330, které má dle NVD CVSS skóre 9,8, s CVSS vektorem CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H [28]. Hodnoty jsou uvedeny v tab. 4.10.

Tab. 4.10: Hodnocení CRLF Injekce dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	N
Rozsah	Nezměněný	U
Dopad na důvěrnost	Vysoký	H
Dopad na integritu	Vysoký	H
Dopad na dostupnost	Vysoký	H

Díky těmto informacím je také možné odvodit hodnocení dle OWAP Risk Rating Methodology. Nalezení a využití zranitelnosti je jednoduché. Zranitelnost je vcelku známá, zároveň je však mnoho automatických nástrojů pro detekci této zranitelnosti v kódu. Díky CVSS je možné zjistit, že zranitelnost má velký dopad na důvěrnost, integritu a dostupnost, a jelikož je možné ji zneužít odkudkoliv přes síť, je možné předpokládat, že útočník je anonymní [28]. Při použití hodnot uvedených v tab. 4.11 a 4.12, je možné zjistit hodnocení závažnosti, které má hodnotu vysoká.

Tab. 4.11: Hodnoty faktorů zranitelnosti pro CRLF

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Jednoduché	7,00
Jednoduchost exploitace	Jednoduché	5,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Aktivní detekce v aplikaci	1,00

Tab. 4.12: Hodnoty faktorů technického dopadu pro CRLF

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhalena všechna data	9,00
Ztráta integrity	Všechna data kompletně znehodnocena	9,00
Ztráta dostupnosti	Všechny služby ztraceny	9,00
Ztráta zodpovědnosti	Anonymní	9,00

4.5 Path Traversal

Path traversal, také známý jako directory traversal, je zaměřen na získání přístupu k souborům či složkám, které jsou mimo složku serveru [29]. Pro využití této zranitelnosti je třeba, aby web, či webová aplikace obsahovala vstup, dle kterého je poté získán soubor. Soubory mimo složku serveru poté mohou být získaný pomocí notace dot-dot-slash (../), která značí pohyb vzhůru v souborovém systému. Pokud není tato zranitelnost ošetřena, je možné získat přístup ke všem souborům na zranitelném zařízení [29].

Nejjednodušší ochranou proti této zranitelnosti je absence vstupu, dle kterého je získán soubor. Další možnosti ochrany mohou být indexace souborů, tedy nepoužívání názvů souborů, ale jen jejich indexů, dle kterých je poté soubor vybrán, ošetření vstupu proti těmto znakům, nebo restrikce, kde se může vyžadovaný soubor nacházet [29].

Hodnocení

Path traversal je v celku jednoduchý útok, který je možné provést přes síť. Nevyžaduje žádná oprávnění, či interakci ze strany uživatele, je však možné získat pouze soubory na zranitelném zařízení. Dopad na důvěrnost dat je vysoký, dopad na integritu a dostupnost však není žádný [29]. Za pomoci těchto údajů je možné získat CVSS skóre, s hodnotou 7,5 a vektorem CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N. Použité hodnoty jsou uvedeny v tab. 4.13.

Tab. 4.13: Hodnocení Path Traversal dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	N
Rozsah	Nezměněný	U
Dopad na důvěrnost	Vysoký	H
Dopad na integritu	Žádný	N
Dopad na dostupnost	Žádný	N

Pro nalezení a využití zranitelnosti jsou dostupné automatické nástroje. Zranitelnost je známá a pro výpočet bude předpokládáno, že útok není nijak logován. Pravděpodobnost tedy bude mít hodnotu 9, tedy vysoká pravděpodobnost. Stejně jako u výpočtu CVSS bude uvažováno, že dopad na důvěrnost bude velmi vysoký, tedy útočník má přístup ke všem datům, ale útok nemá žádný dopad na integritu a dostupnost [29]. Zároveň se dá předpokládat, že jelikož je útok možné provést přes síť, bude útočník anonymní. Pro dopad tedy bude vypočtena hodnota 5, tedy střední. Díky těmto hodnotám je tedy možné získat hodnocení závažnosti rizika zranitelnosti, které má hodnotu vysoká. Hodnoty použité pro výpočet jsou uvedeny v tab. 5.14 a tab. 5.15.

Tab. 4.14: Hodnoty faktorů zranitelnosti pro Path Traversal

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Automatické nástroje k dispozici	9,00
Jednoduchost exploitace	Automatické nástroje k dispozici	9,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Bez logování	9,00

Tab. 4.15: Hodnoty faktorů technického dopadu pro Path Traversal

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhalena všechna data	9,00
Ztráta integrity	Minimální množství málo poškozených dat	1,00
Ztráta dostupnosti	Minimální ztráta sekundárních služeb	1,00
Ztráta zodpovědnosti	Anonymní	9,00

4.6 Denial of Service

Denial of Service (DoS) je zaměřen na způsobení nedostupnosti webových stránek, aplikací či serverů. Existuje několik možností, jak útočník může způsobit úplnou nedostupnost stránek. Může se jednat o útoky zaměřené jak na vytížení zdrojů serveru, na kterém web či aplikace běží, nebo o útok zaměřený na logiku serveru [30]. Pro vytížení zdrojů je většinou užito záplavových útoků. Jedná se o útok, kdy je server, webová stránka či aplikace zahlcena velkým množstvím dat, či požadavků, tudíž je služba pro legitimního uživatele nedostupná. U logických útoků je cíl slabina v protokolu, či aplikaci.

Nejlepší obrana proti těmto útokům je jejich detekce a mitigace. Existuje několik možností, jak tyto útoky detekovat. Jednou z možností je detekce signatur. Tato možnost je často využívaná v bezpečnostních aplikacích, kdy tyto signatury sestavují experti zaměřující se na tuto problematiku. K sestavení signatury je třeba dobrá znalost útoku. Jednou z dalších možností je detekce anomálií, kdy je síťový provoz monitorován. Tato metoda je vhodná pro detekci záplavových útoků, které velmi zvyšují množství provozu, zvyšuje se však také množství falešných poplachů. Mitigace je často realizována pomocí různých firewallů a filtrů.

Hodnocení

Tento útok je možné provést přes síť a komplexita je velmi nízká. Není třeba mít žádné oprávnění a interakce uživatele také není potřebná. Útok je zaměřený primárně na dostupnost serveru či aplikace, ale nemá většinou žádný dopad na důvěrnost či integritu a rozsah je omezený pouze na cíl. Díky těmto informacím je možné určit skóre dle metody CVSS, které má hodnotu 7,5, tedy vysoká závažnost a CVSS vektor CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H [31]. Hodnoty jsou také uvedeny v tab. 4.16.

Tab. 4.16: Hodnocení DoS dle CVSS

Název metriky	Hodnota	Zkratka v CVSS
Vektor útoku	Vnější síťový	N
Komplexita útoku	Nízká	L
Vyžadované oprávnění	Žádné	N
Interakce uživatele	Žádná	N
Rozsah	Nezměněný	U
Dopad na důvěrnost	Vysoký	N
Dopad na integritu	Vysoký	N
Dopad na dostupnost	Vysoký	H

Útok je známý a jsou k jeho provedení dostupné automatické nástroje. Existují však také aplikace pro jeho detekci a mitigaci. U některých typů je také možné vysledovat útočníka. Jak již bylo řečeno dříve, má útok jako hlavní cíl odepřít dostupnost cíle pro legitimní uživatele. Díky těmto informacím je možné Získat hodnocení dle OWAPS Risk Rating Methodology [31]. Hodnoty použité při výpočtu jsou uvedeny v tab. 4.17 a tab. 4.18. Výsledná hodnota závažnosti je poté vysoká.

Tab. 4.17: Hodnoty faktorů zranitelnosti pro DoS

Faktor	Hodnota	Hodnocení
Jednoduchost odhalení	Automatické nástroje k dispozici	9,00
Jednoduchost exploitace	Automatické nástroje k dispozici	9,00
Povědomí	Veřejně známá	9,00
Detekce průniku	Aktivní detekce v aplikaci	1,00

Tab. 4.18: Hodnoty faktorů technického dopadu pro DoS

Faktor	Hodnota	Hodnocení
Ztráta důvěrnosti	Odhaleno minimum necitlivých dat	2,00
Ztráta integrity	Minimální množství málo poškozených dat	1,00
Ztráta dostupnosti	Všechny služby ztraceny	9,00
Ztráta zodpovědnosti	Pravděpodobné vysledování	7,00

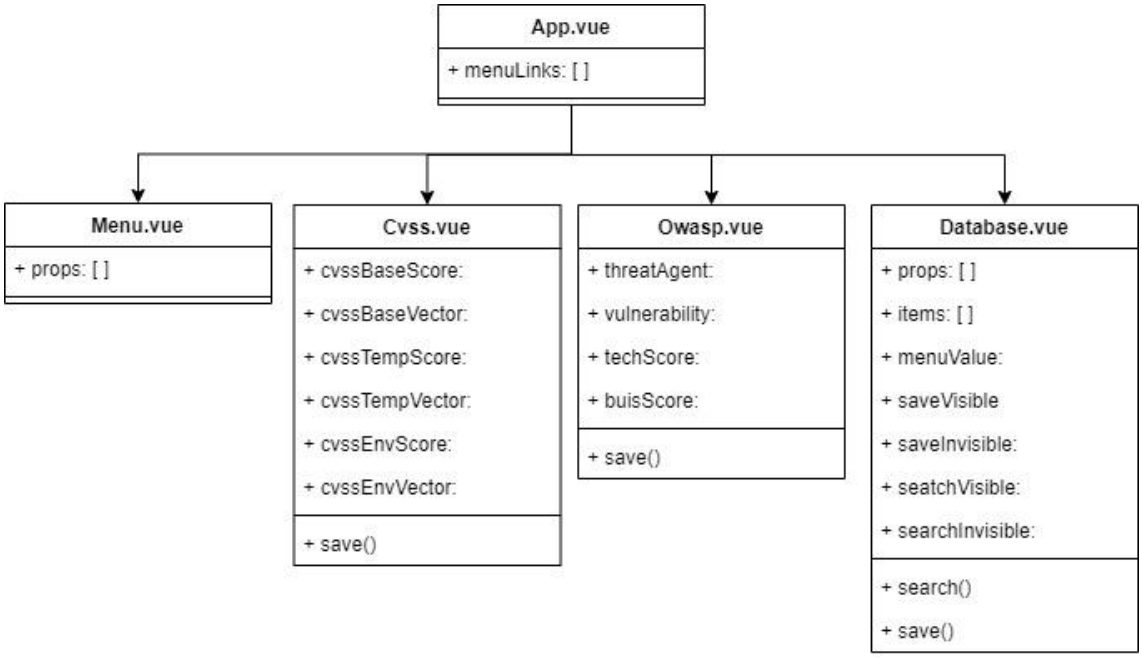
U hodnocení jakékoliv zranitelností je důležité, aby hodnocení vždy prováděla kvalifikovaná osoba. Je třeba mít přehled jak o zranitelnostech samotných, tak i systému, ke kterému se dané zranitelnosti vztahují. Porozumění systému, jeho funkcím a rizikům spojených s úspěšným zneužitím zranitelnosti, může mít společně se znalostí zranitelnosti velký dopad na hodnocení a finální skóre, dle kterého se dále může odvíjet postup odstranění zranitelnosti.

5. NÁVRH APLIKACE

Kalkulátor bude vytvořený za pomoci frameworku Vue.js. Vue.js je framework určený k vytváření grafických uživatelských rozhraní, ale jeho možnosti nejsou omezené pouze na něj [32]. Jednou z hlavních výhod tohoto frameworku je, že HTML kód, JavaScript kód a CSS (Cascading Style Sheets) kód jsou všechny v jednom souboru s příponou .vue.

5.1 Komponenty

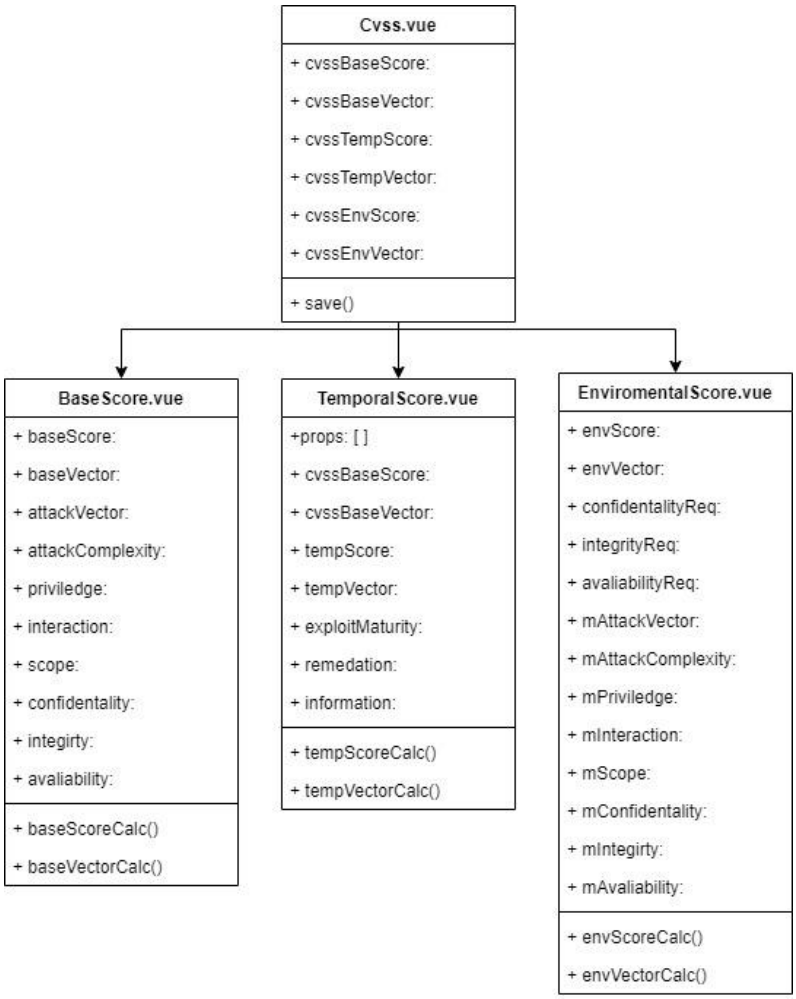
Komponenty jsou hlavní prvky Vue aplikací. Aplikace je tedy poskládaná z různého množství komponentů, které mezi sebou mohou komunikovat a předávat data. V tomto případě tedy samotná aplikace App.vue bude obsahovat komponenty Menu.vue, Cvss.vue, Owasp,vue a Database.vue. Tyto komponenty, krom Menu.vue, které je vždy přítomné, jsou měněny pomocí vue router. Vue router je modul, který umožnuje dynamicky měnit komponenty. Je tedy díky němu možné získat vícestránkovou aplikaci, i když je Vue.js zaměřeno převážně na jednostránkové aplikace. Struktura aplikace je znázorněna v diagramu na obr. 5.1.



Obrázek 5.1: Diagram komponentů App.vue

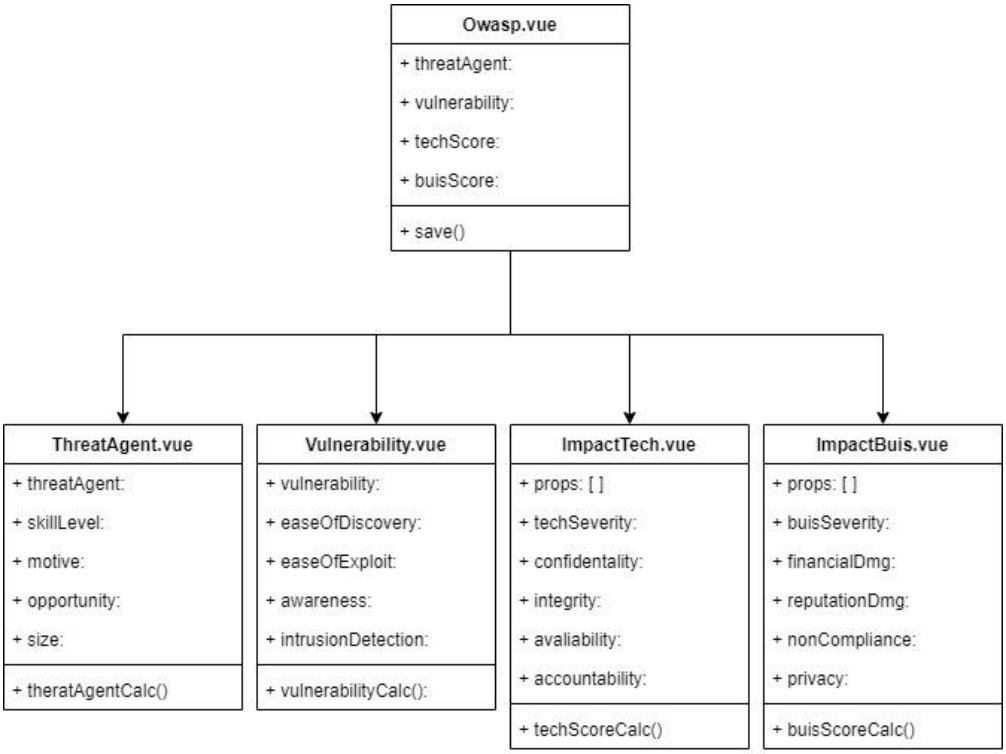
Menu.vue dostává pole objektů z hlavní aplikace. Toto pole obsahuje název a cesty k ostatním komponentům, pomocí kterého je možné ovládat vue router. Je tedy možné přepínat mezi komponenty pomocí menu. Tento komponent je přímo vložen v App.vue, je tedy vždy přítomný, i když se ostatní komponenty změní. Komponent Cvss.vue obsahuje skóre jednotlivých typů CVSS skóre a jejich vektory. Tyto informace jsou

předány z komponentů, které Cvss.vue obsahuje, a v některých případech jsou tyto informace těmto komponentům předány pro výpočet skóre. Komponenty, které Cvss.vue obsahuje, obsahují jednotlivá skóre, data pro výpočet a metody jejich výpočtu. Struktura Cvss.vue je znázorněna v diagramu, který je na obr. 5.2.



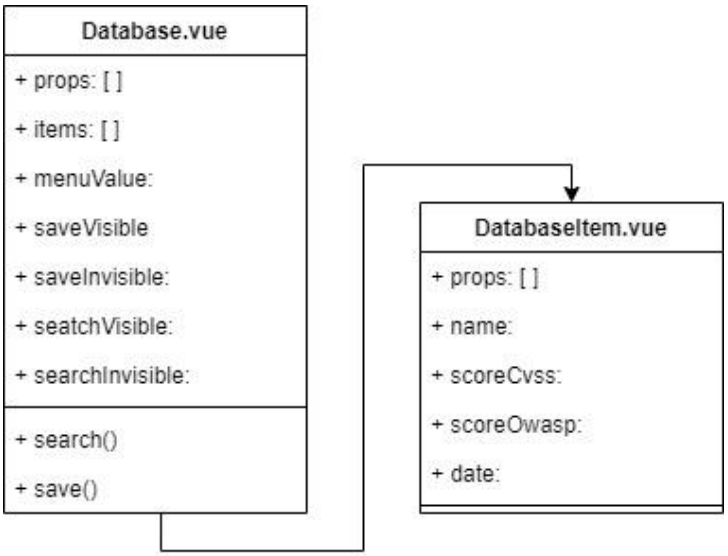
Obrázek 5.2: Diagram komponentů Cvss.vue

Podobným způsobem je také realizován komponent Owasp.vue, který obsahuje výsledky skóre, které jsou předány z komponentů ThreatAgent.vue a Vulnerability.vue, a slovní výsledky, které jsou předány z komponentů ImpactTech.vue a ImpactBuis.vue. Tyto čtyři komponenty, které Owasp.vue obsahuje, obsahují data pro výpočet, a také metody pro výpočet skóre. V diagramu na obr. 5.3 je znázorněna struktura komponentů.



Obrázek 5.3: Diagram komponentů Owasp.vue

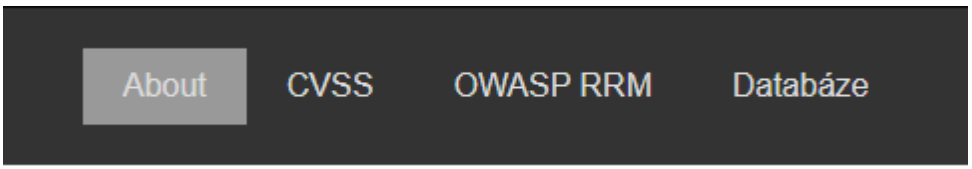
Posledním z možných komponentů, které se mohou měnit pomocí vue router, je komponent Database.vue. Tento komponent obsahuje menu, ve kterém je možné si vybrat, zda chce uživatel do databáze ukládat, nebo v ní vyhledávat. Proto má databáze dva vstupy, jeden pro ukládaní a druhý pro vyhledávání. Databáze také obsahuje komponent DatabaseItem.vue, který je použit pro zobrazení výsledků vyhledávání. Diagram komponentů je zobrazen na obr. 5.4.



Obrázek 5.4: Diagram komponentů Database.vue

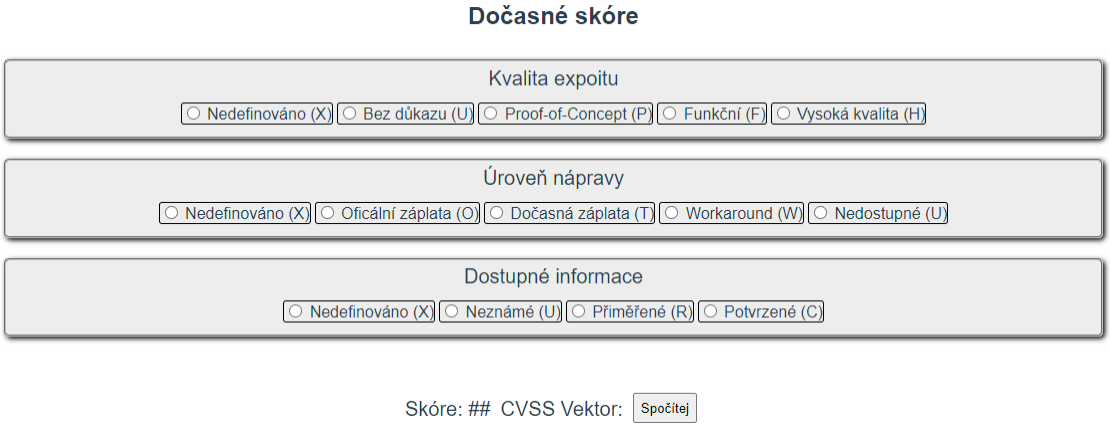
5.2 Návrh grafického uživatelského rozhraní

Je třeba vytvořit vzhled menu, které bude použito pro navigaci v aplikaci. Názvy budou přidány do seznamu, a jednotlivé názvy budou reprezentovat samotná tlačítka k ovládání vue router. Menu je zobrazeno na obr. 5.5.



Obrázek 5.5: Menu s jednotlivými odkazy s kurzorem nad odkazem About.

Po vytvoření návrhu menu je třeba vytvořit návrh komponentů Cvss.vue a Owasp.vue. Tyto komponenty mají v podstatě stejný vzhled, je tedy možné použít stejný styl pro oba komponenty. Pro samotný výběr hodnot faktorů budou použity přepínače. Přepínače faktorů budou seskupeny pomocí fieldestů, a všechny fieldsety jednotlivých komponentů budou seskupeny pomocí sekcí (section). Samotným fieldsetům bude přidána hranice, zaoblení rohů a stín. U komponentů Cvss.vue budou také navíc přidány výstupy pro zobrazení skóre a vektoru jednotlivých typů skóre. Jeden z komponentů Cvss.vue je zobrazen na obr. 5.6.

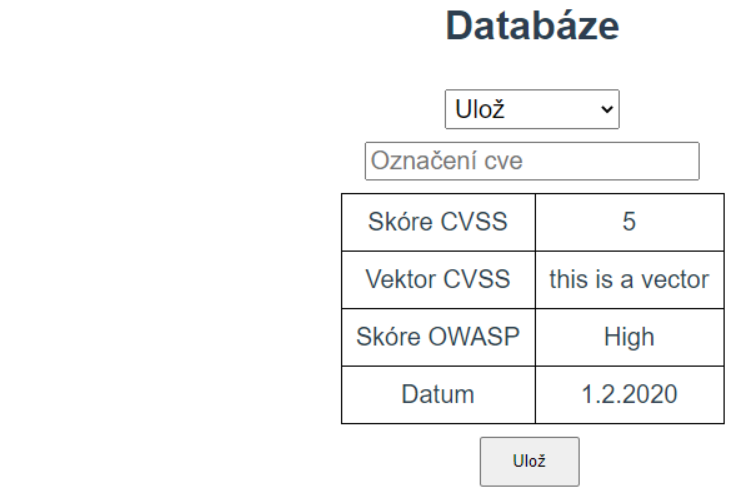


Obrázek 5.6: Část stránky s hodnocením CVSS

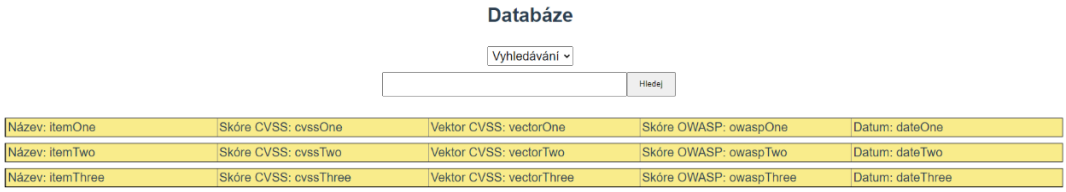
K jednotlivým přepínačům budou přidány štítky a pomocí vložení přepínačů do štítků bude dosaženo toho, že samotný přepínač bude zobrazen uvnitř štítku. Dále bude také přidána změna pozadí štítku v případě, že se nad ním nachází kurzor. Toto je zobrazeno na obr. 5.7. Veškeré toto nastavení stylů je pouze v komponentech Cvss.vue a Owasp.vue.

Obrázek 5.7: Část dočasného skóre s kurzorem na přepínačem Bez důkazu (U)

Pro databázi bude přidáno menu pro výběr akce. Dle výběru je poté zobrazeno rozhraní pro ukládání, či vyhledávání v databázi. Při ukládání bude zobrazen textový vstup pro název zranitelnosti a momentální vypočtené hodnoty, zatímco při výběru vyhledávání bude zobrazen textový vstup pro zadání řetězce pro vyhledání. Obě možnosti jsou zobrazeny na obr. 5.8 a obr. 5.9.



Obrázek 5.8: Databáze při výběru možnosti Ulož



Obrázek 5.9: Databáze při výběru možnosti Vyhledávání

5.3 Backend

Jelikož aplikace vyžaduje ukládání a vyhledávání výsledků, je třeba použít databázi, která bude tato data ukládat. Pro databázi je zvolena databáze PostgreSQL. Framework Vue.js však není schopný komunikovat přímo s databází. Je proto třeba vytvořit API pro komunikaci mezi frontendem a databází. Pro API je zvolen framework Django a Django REST (Representational State Transfer) framework, který je určen k implementaci webových API [33]. V samotné databáze bude třeba vytvořit novou prázdnou databázi, se kterou bude poté vytvořená API komunikovat. Pro komunikaci bude třeba vytvořit v API nový model, který obsahuje informace, které je třeba uložit. Jedná se o název, skóre dvou metodik hodnocení, CVSS vektor a datum. Tyto informace budou sloupce v automaticky vytvořené tabulce v databázi.

6. IMPLEMENTACE

Jelikož budou data ukládána do databáze, je třeba vytvořit dvě aplikace, a to pro backend a frontend. I když v podstatě nezáleží na pořadí implementace, je vhodné začít s implementací backendu, kde je třeba naimplementovat komunikaci s databází a API pro frontendovou aplikaci. Následně ve frontendové aplikaci proběhne implementace samotných výpočtů a vzhledu aplikace, které budou komunikovat s backendem.

6.1 Backend

Jak již bylo řečeno v předchozí kapitole, pro backend bude použita database PostgreSQL, a pro API framework Django a Django REST framework. Jako první je tedy potřeba naistalovat python. Z technických důvodů je lepší dále pracovat ve virtuálním prostředí. Pro tuto aplikaci bude použito python virtuální prostředí, virtualenv. Virtuální prostředí bude vytvořeno pomocí příkazu *virtuelenv <název>*. Následně je třeba nainstalovat framework Django, Django REST framework, django-cors-headres, django-csp a psycopg2, který bude použit pro komunikaci s PostgreSQL. Po instalaci všeho potřebného bude vytvořen nový Django projekt a následně nová aplikace.

V PostgreSQL je třeba vytvořit novou prázdnou databázi. Je možné databázi vytvořit z příkazové řádky, nebo pomocí aplikace pgAdmin. Dále už je třeba pracovat ve vybraném IDE (Integrated Development Enviroment). Nejdříve budou provedeny změny v souboru pro nastavení. Zde se musí nově vytvořená aplikace, djagno-cors-headers (CORS, Cross-Origin Resource Sharing) a REST framework vložit do části *INSTALLED_APPS* v souboru *settings.py*. Následně bude vytvořena pod sekcí *ALLOWED_HOSTS* sekce *CORS_ALLOWED_ORIGINS*, kde proběhne nastavení povolených hostů. Pro nastavení databáze na PostgreSQL je třeba změnit část *DATABASES*. Zde bude změněna část *ENGINE*, kde stačí pouze přepsat defaultní databázi na *postgresql* a název databáze *NAME*, kam bude vložen název vytvořené databáze. Dále je v této časti nutné vytvořit nové položky *USER* a *PASSWORD*, do kterých budou vloženy přihlašovací údaje k databázi, a *HOST*, kam bude vložena adresa, na které databáze běží. Následně je třeba vytvořit superuživatele příkazem *manage.py createsuperuser*. Tento uživatel bude použit k přístupu k administrátorským funkcím API. Pro uložení těchto změn je třeba migrovat defaultní nastavení a tabulky pomocí příkazu *manage.py migrate*, poté by měly v databázi být nově vytvořené tabulky.

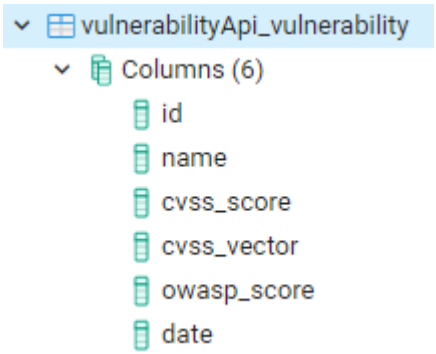
Po změně základního nastavení třeba nastavit některá bezpečnostní opatření. Nejdříve bude provedeno nastavení CSP (Content Security Policy) pro backend. Jedná se především o nastavení zdroje stylů, skriptů, fontů a obrázků na hodnotu *self*. Tím je zajištěno, že se nespustí žádný škodlivý kód, který nepochází z aplikace samotné.

Modely se konfigurují v souboru *models.py*. V tomto souboru bude vytvořena třída, která dědí z třídy *Models*. Do této třídy budou vloženy vlastnosti (properties), které budou přímo korespondovat se sloupci v tabulce v databázi [33]. Jak již bylo řečeno v návrhu, je třeba mít pět vlastností, které budou použity. Jedná se o název, skóre obou hodnocení, CVSS vektor a datum. Pro tyto vlastnosti je zadán jejich název a typ, v tomto případě *Charfiled*. Jelikož je použit *Charfiled*, je vhodné také zadat maximální délku řetězce. Ta je nastavena u CVSS skóre, vektoru, OWASP skóre a data na jejich maximální možné hodnoty, zatímco název bude ponechán s poněkud větší délkou.

Výpis 6.1: Vytvoření modelu

```
#vytvoření modelu pro databázi
Class Vulnerability(models.Model):
      name=models.Charfield(max_length=40, default='-')
      cvss_score=models.CharField(max_length=3, default='-')
      cvss_vector=models.CharField(max_length=110, default='-')
      owasp_score=models.CharField(max_length=7, default='-')
      date=models.CharField(max length=10, default='-')
```

Po vytvoření modelu je třeba opět provést migraci, tentokrát však dvěma příkazy, a to *manage.py makemigrations <jméno aplikace>*, a poté už opět *manage.py migrate* [33]. Následně by měla databáze obsahovat novou prázdnou tabulku se sloupci korespondujícími s modelem, viz obr.6.1. Dále je třeba tento model naimportovat a registrovat v souboru *admin.py*, aby bylo možné upravovat záznamy v databázi.



Obrázek 6.1: Vytvořená tabulka modelu v databázi

Po vytvoření modelu je třeba vytvořit serializátor (Serializer). Ten slouží k převedení komplexích dat do dat typu JSON (JavaScript Object Notation), nebo XML (Extensible Markup Language), a zpět. Nejdříve bude vytvořen nový soubor ve složce aplikace s názvem *serializers.py*, v tomto souboru bude naimportován modul *serializers* a vytvořený model. Následně bude vytvořena třída serializátoru. Dobrým zvykem je nazvat tuto třídu *<název modelu>Serializer*. Tato třída bude dědit z třídy *ModelSerializers*. Dále bude v této třídě vytvořena nová třída *Meta*, ve které budou vytvořeny dvě proměnné, a to *model*, která koresponduje s vytvořeným modelem a *fields*, ve kterém bude určeno, jaké proměnné budou převedeny [33].

Nyní budou vytvořeny tzv. views v souboru *views.py*. Nejdříve budou naimportovány moduly *api_view* a *Response*, dále už vytvořený model a serializátor. *Api_view* umožňuje určit jaké metody mohou být použity a *Response* je metoda, která předává data. Dále je třeba definovat metody pro zobrazení, uložení a vyhledávání, kdy pro zobrazení a vyhledávání bude použita metoda GET, a pro uložení metoda POST. Tyto definované metody přijímají argument *request* a v případě vyhledávání také vyhledávaný řetězec. V metodě pro zobrazení všech prvků v databázi, vulnerabilityList, budou vytvořeny dvě proměnné. První proměnná *vulnerabilities* bude obsahovat všechny objekty vytvořeného modelu, zatímco druhá proměnná *serializer* bude obsahovat serializovaný obsah proměnné *vulnerabilities*. Obsah se serializuje pomocí vytvořeného serializátoru, který přijímá dva argumenty. První z nich proměnná, která bude serializována, a druhý, zda bude serializována jedna, či více položek. Dále data proměnné *serializer* budou vráceny pomocí metody Response, kde jako další z argumentů bude *headers*, kde bude přidána hlavička *Content-Disposition* [33]. Ta obsahuje typ předávaných dat.

Výpis 6.2: Metoda pro získání zranitelností z databáze

```
1  @api_view(['GET'])
2  def vulnerabilityList(request):
3        vulnerabilities=Vulnerability.objects.all() #Vyzvednutí
   všech objektů
4        serializer=VulnerabilitySerializer(vulnerabilities,
   many=True) #Sewrializace objektů
5        return Response(serializer.data, #Vracení dat v odpovědi
6        headers={'Content-Disposition': 'attachment;
   filename="api.json"' #Konfigurace dalších hlaviček
```

U metody pro vyhledávání, vulnerabilitySearch, je postup podobný, je zde však navíc argument, který obsahuje hledaný řetězec. V tomto případě stačí proměnnou *vulnerabilities* filtrovat pomocí tohoto řetězce a pomocí *Response* vrátit pouze položky, které vyhovují. Pro metodu ukládání už nebude použita metoda GET, ale POST. Proměnná *serializer*, bude obsahovat serializovaná data *requestu*. Dále, pokud bude proměnná *serializer* validní, tak bude uložena metodou *save( )*.

Tyto tři metody stačí pro funkčnost s frontendovou aplikací. Pro lepší přehled a informace o API je však vhodné vytvořit ještě jednu metodu, která bude zopbrazovat všechny možné metody a URL. Bude tedy vytvořena metoda vulnerabilityOverview, která bude používat metodu GET. V této metodě bude vytvořen JSON, který bude obsahovat název a URL k metodě. Tento JSON bude vrácen pomocí metody Response. K přístupu k jednotlivým *views* je potřeba vytvořit URL cesty. Pro tyto cesty bude vytvořen nový soubor urls.py ve složce aplikace. Zde bude naimportován path z django.urls a vytvořený soubor views. Dále bude v tomto souboru vytvořeno pole *urlpatterns*, do kterého budou vloženy funkce path( ), které přijímají tři argumenty, a to cestu, view, ke kterému daná cesta patří a název. Pro použití těchto cest je však třeba je uvést do souboru urls.py ve složce projektu. V tomto souboru je potřeba upravit druhý import, kde je importován modul *path*, kam bude přidán další modul *include*. Dále do *urlpatters* bude uvedena další cesta, v tomto případě se souborem s vytvořenými cestami [33].

Nyní už je možné API spustit příkazem *python manage.py runserver*. Po přístupu na adresu, na které API běží, lze přidat k URL řetězec /api/, a měl by být zobrazen dříve vytvořený seznam přístupných adres, jako je možné vidět na obr. 6.2.

Obrázek 6.2: Seznam možných adres v API

Dále je možné API otestovat přístupem na všechny možné URL adresy, případně vložením testovacích údajů, které musí být ve formátu JSON. Pokud ukládání funguje, měly by být tyto data zobrazeny jak v části API pro zobrazení všech uložených dat, tak i v samotné databázi. Data zobrazená v API jsou na obrázku obr. 6.3.

Obrázek 6.3: Data zobrazená v API

6.2 Frontend

Aplikace bude napsána ve frameworku Vue.js, je ho tedy třeba nejdříve naistalovat. Jelikož Vue.js potřebuje k instalaci Node.js, je třeba Node nainstalovat jako první. Po instalaci Node, je Vue.js naistalován příkazem *npm install vue*. Dále už je možné vytvořit samotnou aplikaci pomocí příkazu *vue create <název aplikace>*. Název aplikace nesmí obsahovat velká písmena. Pro vytvoření aplikace bude vybráno nastavení *Manually select features* pro vytvoření projektu. V následujícím menu bude vybrána možnost *Router* a možnost *y* pro *history mode*. Následně může být projekt vytvořen. Po vytvoření projektu je třeba naistalovat poslední potřebný modul, axios. Dále už bude práce probíhat v preferovaném IDE. Jako první bude zajištěna komunikace pomocí HTTPS (Hypertext Transfer Protocol Secure). Bude vytvořen konfigurační soubor *vue.config.js*, ve kterém bude povoleno HTTPS. Lze jej aktivovat pomocí nastavení *https* na hodnotu *True*, nebo lze přímo uvést cesty k jednotlivým certifikátům [32].

Vue router

Je třeba vytvořit hlavní komponenty aplikace. To budou rodičovské komponenty pro dvě metody hodnocení, jeden komponent pro databázi, a případně jeden komponent pro základní informace o aplikaci. Tyto komponenty by měly být vytvořeny ve složce *views*. Složka *views* by měla obsahovat komponenty, které budou zobrazeny pomocí routeru a budou tedy mít vlastní cesty. V nově vytvořených komponentech bude vytvořen tag *<template>*, který bude obsahovat HTML. Pro test budou napsána jména jednotlivých komponentů. Dále, v souboru *index.js* ve složce *router*, budou naimportovány všechny vytvořené komponenty a bude vymazán obsah konstanty *routes*. Do tohoto pole budou následně vložena vlastní data, tedy cesty, názvy a samotné komponenty.

Výpis 6.3: Konfigurace cesty v souboru index.js routeru

```
const routes=[
      {
       path:'/', //cesta URL ke stránce
       name:'About', //Název stránky
       component:About //Komponent stránky
      },
      ]
```

V komponentu *App.vue* jsou dvě možnosti, kterými je možné vypsat cesty pro komponenty. Buď obdobně jako už existující cesty budou vytvořeny další, pro nové komponenty, nebo bude použita funkce Vue.js, a bude vytvořena smyčka, která postupně všechny cesty vypíše. V tomto případě je zvolena druhá možnost. Nejdříve bude vytvořena část <*script*>, ve které bude deklarováno pole, které bude obsahovat cestu a zobrazený název. Tento seznam bude použit pro smyčku for, která bude vypisovat všechny cesty. Bude vytvořen tag <*nav*>, který bude obsahovat navigační linky a dále bude vytvořen neřazený seznam, ve kterém budou pomocí funkce *v-for* vypsány všechny cesty. *V-for* používá speciální syntax, ve formě *položka in položky*, kde *položky* je pole [32]. Lze tedy použít *v-for* pro vypsání několika položek seznamu, které budou obsahovat jednotlivé linky na komponenty.

Výpis 6.4: Vytvoření odkazů na jednotlivé části

```
<nav>
      <ul>
            <li v-for="(link, index) in menuLinks" :key="index">
                  <router-link :to="link.path">
                        {{ link.text }}
                  </router-link>
            </li>
      </ul>
</nav>
```

Dále je možné vymazat nepotřebné komponenty. Dále bude spuštěna samotná aplikace pomocí příkazu *npm run serve*. V příkazovém řádku je uvedena adresa, na které aplikace běží, po otevření této adresy je možné vidět celkově čtyři odkazy na vytvořené komponenty, viz obr. 6.4. Jednou z výhod frameworku Vue.js je fakt, že server je možné nechat spuštěný a zároveň provádět změny aplikace, které se zobrazují v reálném čase ihned po uložení.

Obrázek 6.4: Odkazy na komponenty v prohlížeči

Po otestování všech odkazů, stačí pouze upravit vzhled. Ten jde změnit dvěma způsoby. Buď přímo v tagu, nebo je upravena část <*style*>, do které je možné psát jako do klasického CSS souboru. Část *style* také může obsahovat parametr *scoped*, který určuje, zda se tento styl bude aplikovat pouze na tento komponent. V opačném případě by se vztahoval na celou aplikaci.

Common Vulnerability Scoring System

Jako první je třeba vytvořit složku pro komponenty CVSS skóre ve složce *components*. Dále budou v nově vytvořené složce vytvořeny tři komponenty pro tři druhy skóre. Jako první bude implementován komponent pro základní skóre. V tomto komponentu bude vytvořena základní struktura, tedy *template*, *script* a *style*. Jelikož *template* může obsahovat pouze jeden prvek, je třeba vytvořit jeden prvek, který bude obsahovat všechny ostatní, např. div. Do tohoto prvku už je možné vkládat libovolné množství jiných prvků. Bude tedy vytvořen nadpis pro název skóre, div pro rozdělení výběru možností a dále tlačítko pro výpočet a uložení. Dále budou v nově vytvořené položce div vytvořeny fieldsety, které, jak bylo řečeno v návrhu, seskupují přepínače pro jednotlivé faktory. Do těchto fieldsetů budou vloženy názvy (legend) a přepínače a jejich štítky (label). Přepínače budou mít dále hodnoty jako písmena dle CVSS, která budou později také využita pro vytvoření vektoru [4]. Po vytvoření všech přepínačů a štítků, bude vytvořeno tlačítko pro spočítání skóre a výstup pro zobrazení skóre. Nejdříve je třeba vytvořit div pro tlačítko a výstupy. V něm budou vloženy dva paragrafy pro skóre a vektor, a jedno tlačítko pro výpočet skóre. Dále je třeba tento komponent naimportovat do komponentu *Cvss.vue*, aby bylo možné jej zobrazit. V komponentu *Cvss.vue* v části *script* bude naimportován tento komponent a dopsán defaultní export tohoto objektu, do kterého bude tento komponent vložen [32]. Následně je možné komponent použít jako běžný HTML tag. Po použití bude tento komponent zobrazen v prohlížeči.

Výpis 6.5: Import komponentu

```
<script>
import BaseScore from '../components/cvss/BaseScore.vue' //Import
komponentu
export default {
      components:{
            BaseScore //Export komponentu
      }
}
</script>
```

Dále je třeba vytvořit metody pro samotný výpočet skóre. Jako první budou deklarovány potřebné proměnné, které budou použity. Jedná se o proměnné pro všechny metriky základního skóre, dále pro skóre samotné a vektor. Všechny proměnné, krom skóre a vektoru, budou nastaveny na výchozí hodnotu null. Skóre a vektor budou mít výchozí hodnotu prázdný řetězec. Tyto proměnné budou deklarovány ve funkci *data( )*. Aby bylo možné svázat proměnné a přepínače, bude využita direktiva *v-model* [32]. K tagům input bude přidán *v-model* a určeno, ke které proměnné je svázán. Metody se vytváří pod funkcí data( ), kde bude vytvořena část *methods:*, která bude všechny metody obsahovat. Jako první bude vytvořena metoda pro výpočet skóre. Nejdříve budou v této funkci deklarovány proměnné pro všechny metriky, dále pro sub skóre dopadu, dopad, sub skóre základního skóre, základní skóre, exploitovatelnost a pro základní faktory. Proměnná pro základní faktory bude použita později pro přenos hodnot metrik základního skóre do skóre prostředí. Pro určení hodnot jednotlivých metrik bude použit přepínač switch case. S jeho pomocí budou k jednotlivým hodnotám přepínačů určeny číselné hodnoty. Je třeba aby byl tento přepínač (switch case) upraven pro faktor požadované oprávnění, který se mění s rozsahem. To lze vyřešit pomocí podmínky. Poté už je možné implementovat samotný výpočet. Nejprve je třeba vypočíst exploitovatelnost a sub skóre dopadu. Rovnice pro výpočty jsou uvedeny jak v předchozí kapitole, tak i v dokumentaci [4]. Pomocí podmínky bude určeno, zda se jedná o výpočet se změněným, či nezměněným rozsahem. Použité funkce Mininimum a Roundup lze nahradit. Funkce Minimum pomocí podmínky a funkci Roundup pomocí Math.ceil( ), která zaokrouhlí dané číslo na vyšší celé číslo. Jelikož je třeba zaokrouhlit na jedno desetinné místo, bude nejprve výsledek vynásoben deseti, a po průběhu funkce opět vydělen. Po výpočtu bude hodnota základního skóre nastavena na vypočtenou hodnotu. Dále bude zavolána metoda pro vytvoření vektoru a všechny potřebné proměnné budou předány rodičovskému komponentu pomocí funkce *$emit* [32]. Tato funkce přijímá dva argumenty, a to název, pod jakým proměnnou vyzvedne rodičovský komponent, a samotnou proměnnou, která bude předána. Následně zavolána funkce pro uložení v rodičovském komponentu [32]. Tím se skóre uloží ihned po výpočtu.

Funkce pro vytvoření vektoru je vcelku jednoduchá. Je třeba pouze vytvořit řetězec, který bude obsahovat páry zkratek metrik a hodnot z přepínačů. Tento řetězec bude poté uložen do proměnné vektoru. Po vytvoření této funkce je výpočet základního skóre a vektoru funkční. Jelikož byla předána některá data rodičovskému komponentu, je třeba je také vyzvednout. Pro vyzvednutí budou potřebné proměnné pro tyto data v rodičovském komponentu. V komponentu *Cvss.vue* bude potřeba celkem deset proměnných. Šest pro skóre a vektory jednotlivých výpočtů, jedna pro hodnoty metrik základního skóre a tři pro kvalitu exploitu, úroveň nápravy a důvěrnost reportu. Dále je potřeba předat data dětskému komponentu [32]. Tyto data je možné vyzvednout v dětském komponentu pomocí *props*. *Props* pole, díky kterému je možné vyzvednout proměnné předané z rodičovského komponentu [32].

Výpis 6.6: Předání dat z a do dětského komponentu

```
1  <!--Vyzvednutí dat z dětského komponentu-->
2  <detskyKomponent
   @<nazevVDetskemKomponentu>="<nazevVRodicovkemKomponentu>=$event">
   </detskyKomponent>
3  <!--Předání dat dětskému komponentu-->
4  <detskyKomponent
   :<nazevVDetskemKomponentu>="<nazevVRodicovskemKomponentu>">
   </detskyKomponent>
```

V komponentu pro dočasné skóre je u HTML části postup v podstatě stejný jako v předchozím komponentu. V části *script* však třeba vyzvednout data z rodičovského komponentu. Pro výpočet jsou potřeba jak základní skóre, tak i vektor. Jak již bylo popsáno, tyto data budou předány z rodičovského komponentu a vyzvednuty pomocí *props*. Dále bude vytvořeno pět proměnných pro skóre, vektor a tři metriky dočasného skóre. Metody pro výpočet dočasného skóre budou implementovány stejným způsobem, jako v předchozím komponentu. Je třeba nejdříve deklarovat proměnné a poté pomocí switch case určit hodnoty přepínačů. Následně bude dle rovnic vypočteno skóre a zavolána metoda pro vytvoření vektoru a tyto data budou předána rodičovskému komponentu [4]. Jelikož skóre prostředí vyžaduje tři metriky z tohoto skóre, budou také předány rodičovskému komponentu. Metoda pro vytvoření vektoru bude, jako i v předchozím případě, pouze vytvoření řetězce s vektorem. Dále bude tento komponent naimportován do komponentu *Cvss.vue,* stejně jako předchozí komponent.

V komponentu pro skóre prostředí je postup stejný, jako v předchozích případech. Budou vytvořeny přepínače pro metriky, zobrazení skóre, vektoru a tlačítko pro výpočet. Jelikož jsou pro výpočet potřebné dočasné skóre a vektor, a také kvůli modifikovaným faktorům všechny faktory základního a dočasného skóre, budou tyto hodnoty předány z rodičovského komponentu. Po vyzvednutí a deklarování všech potřebných proměnných, budou implementovány metody výpočtu. Metody mají podobný princip jako u předchozích komponentů, zde je však třeba nastavit hodnoty modifikovaných faktorů na hodnoty ze základního skóre v případě, že je modifikovaný faktor nedefinován.

Toho je možné dosáhnout pomocí podmínky a nastavení hodnoty z pole, které bylo předáno z komponentu základního skóre. Problém nastává v případě změny rozsahu, jelikož existují faktory, které se mění v závislosti na něm. V případě že se tyto faktory změnily ve skóre prostředí, není třeba provádět žádné změny, a lze použít stejný postup jako u základního skóre. V případě že se však tyto faktory nezměnily, je třeba změnit jejich hodnotu v závislosti na modifikovaném rozsahu. Výpočet skóre bude implementován dle rovnic [4]. Na konci metody výpočtu bude zavolána funkce pro výpočet vektoru a skóre s vektorem budou předány rodičovskému komponentu Následně bude zavolána metoda pro uložení. Funkce pro výpočet tohoto vektoru má jinou formu než ostatní. Ve skóre prostředí se faktor nepřidává do vektoru, pokud má hodnotu „nedefinováno“. Budou tedy vytvořena dvě pole, *factors* se zkratkami metrik, a *values* s proměnnými, které jsou svázány s přepínači. Do pole *values* budou zapsány momentální hodnoty přepínačů. Dále je třeba tyto pole pomocí smyčky for projít s podmínkou, že pokud hodnota v poli *values* bude definována, tedy nemá hodnotu X, bude do proměnné s vektorem zapsána zkratka faktoru a hodnota. Tato proměnná bude následně přidána k vyzvednutému vektoru a tím bude získán vektor skóre prostředí.

Poslední metoda, kterou je třeba implementovat, je metoda pro uložení v komponentu *Cvss.vue*. Bude tedy vytvořena metoda *save( )*, která rozpozná které skóre je vypočítáno a podle toho bude předáno dané skóre rodičovskému komponentu. Pro rozpoznání skóre bude použita podmínka, která kontroluje, zda má proměnná nějakou hodnotu. V komponentu *App.vue* musí být tyto data vyzvednuta.

Open Web Application Security Project Risk Rating Methodology

Podobně jako u předchozí metody výpočtu bude vytvořena složka a v tomto případě čtyři potřebné komponenty. Jedná se o komponenty pro faktory nositele hrozby, faktory zranitelnosti a faktory technického dopadu a dopadu na byznys. Dále tyto komponenty budou naimportovány do komponentu *Owasp.vue*, což je komponent vytvořený pro OWASP Risk Rating Methodology. V tomto komponentu budou vytvořeny proměnné pro hodnoty faktorů nositele hrozby, zranitelnosti a dvě proměnné výsledného skóre dle typu použitého dopadu.

Jako první bude naimplementován komponent pro faktory nositele hrozby. Podobně jako u komponentů skóre CVSS bude vytvořena struktura a HTML část komponentu. Dále budou vytvořeny potřebné proměnné a vazby s přepínači pomocí *v-model* [32]. Následně bude vytvořena metoda pro výpočet. Výpočet je mnohem jednodušší než u CVSS skóre, jelikož se jedná pouze o algebraický průměr hodnot [12]. V metodě budou vytvořeny proměnné, do kterých budou vloženy hodnoty pomocí switch case, jako v případě CVSS skóre. Poté budou tyto hodnoty sečteny a bude vypočten algebraický průměr. Toto číslo není nijak neupravována a bude předáno rodičovskému komponentu pomocí funkce *$emit* [32]. V tomto komponentu nebude vytvořeno tlačítko pro výpočet,

jelikož tato část skóre je pouze potřebná pro výpočet celkového skóre. Toto skóre není možné využít jako samostatné hodnocení, je tedy vhodné, aby funkce pro výpočet byla volána automaticky. Funkci pro výpočet zavolá buď funkce pro výpočet skóre pomocí technického dopadu, nebo funkce pro výpočet pomocí dopadu na byznys.

V komponentu *Vulnerability.vue* budou vytvořeny přepínače a proměnné, které budou svázány s přepínači pomocí *v-model*, a vytvořena metoda pro výpočet. Výpočet bude v podstatě stejný jako v předchozím komponentu, a i zde se jedná pouze o část celkového skóre, nebude tedy dostupné tlačítko pro výpočet.

V komponentu *TechImpact.vue* bude třeba naimplementovat přepínače, vyzvednout předaná data a vytvořit proměnné. Bude také vytvořen výstup pro skóre a tlačítko pro výpočet, které po stisknutí zavolá danou metodu v rodičovském komponentu. V metodě pro výpočet technického dopadu budou, jako v předchozích komponentech, získány hodnoty přepínačů a vypočten algebraický průměr. Dále však bude vypočtena hodnota pravděpodobnosti zneužití, která nabývá hodnot 1–3. Hodnota bude určena vytvořením průměru ze skóre nositele hrozby a zranitelnosti, které je dále převedeno podle výsledného čísla na hodnoty 1–3 pomocí podmínky [12]. Stejným způsobem bude převedeno na hodnoty 1–3 také skóre dopadu. Poté bude spočteno výsledné hodnocení sečtením pravděpodobnosti a dopadu. Výsledná hodnota bude převedena na kvalitativní hodnocení pomocí podmínky a určených hodnot [12]. Toto hodnocení bude následně předáno rodičovskému komponentu. V komponentu *BuisImpact.vue*, bude postupováno v podstatě stejným způsobem, jako u tohoto komponentu. Jediným rozdílem jsou faktory a názvy proměnných.

Jakmile je dokončena implementace těchto dvou komponentů, je třeba vytvořit dvě metody v komponentu *Owasp.vue*. Jedná se o metody pro výpočet skóre dle technického a byznys dopadu. Jelikož tato metoda bude v rodičovském komponentu, bude třeba volat metody v dětských komponentech. Pro tuto funkcionalitu bude využito referencí [32].

Výpis 6.7: Reference komponentu

```
1 <!--reference komponentu -->
2 <ThreatAgent ref="ThreatAgent></ThreatAgent>
```

Díky nim je možné z rodičovského komponentu přímo použít funkce z dětského komponentu. Metoda v rodičovském komponentu tedy zavolá metody v dětských komponentech, a tím vypočítá skóre. Jako poslední bude třeba vytvořit funkci pro uložení. Tato funkce, stejně jako funkce pro CVSS, však skóre neuloží přímo do databáze, ale jen předá skóre rodičovskému komponentu *App.vue*, který jej později předá komponentu *Database.vue*. Předané skóre bude určeno dle toho, které skóre bude vypočteno s tím, že bude preferováno skóre dle dopadu na byznys.

Databáze

V komponentu *Database.vue* bude vytvořen formulář, který bude obsahovat tabulku, která zobrazí název, momentálně vypočítané skóre, vektor a datum. Pro název bude jedna buňka tabulky obsahovat textový vstup. Dále bude v tomto formuláři vytvořen textový vstup pro název a tlačítko, které data z tohoto formuláře pošle backendu. Toto tlačítko však nebude typu button, ale submit. Vytvořený formulář bude sloužit k ukládání a zavolá pomocí *v-on:submit* metodu pro uložení. Rodičovský komponent *App.vue* obsahuje router-view. I tento tag je také pouze komponent, je tedy možné předávat data stejným způsobem, jako u jiných komponentů. Skóre a vektor budou tedy předány dětskému komponentu a vyzvednuty v komponentu *Databse.vue*. Dále budou vytvořeny proměnné pro název a datum, a pole, které bude později obsahovat data předaná z backendu. Jelikož bude zranitelnost uložena i s datem, je třeba vytvořit nové datum, a z něj získat den, měsíc a rok. Aby bylo datum vytvořeno automaticky, budou funkce pro získání aktuálního data uvnitř metody mounted( ), která proběhne před renderováním samotného komponentu. Dále bude vytvořena funkci pro uložení dat do databáze.

Jelikož ke komunikaci mezi frontendem a backendem bude použit axios, je třeba ho nejdříve naimportovat. Vytvořené API přijímá data ve formátu JSON. Pro uložení je tedy třeba vytvořit data v tomto formátu, která budou předána pomocí metody POST backendu. Tato metoda přijímá dva argumenty, URL adresu a proměnnou typu JSON. Odpověď této metody bude obsahovat ukládaný prvek, a data z této odpovědi budou přidána do vytvořeného pole pro zranitelnosti. V případě chyby, bude chyba vypsána do konzole. Pro získání dat z backendu bude vytvořena metodu, která bude používat metodu GET. Jelikož by data měla být získána z databáze před načtením samotného komponentu, bude zavolána z metody created( ). V této metodě bude tedy získána odpověď a její data vložena do dříve vytvořeného pole [34]. V případě, že databáze není funkční, bude vypsána chyba do konzole a uživateli bude zobrazena chybová hláška s informací, že databáze momentálně nefunguje.

Výpis 6.8: Získání dat z databáze

```
getItems(){
      axios
            .get(http://127.0.0.1:8000/api/vulnerability-list,
             {timeout:1000}) //GET s URL a Timeoutem
            .then((res)    =>    (this.vulnerabilities=res.data))
//vložení dat do připraveného pole
            .catch((err) => {alert('Chyba databáze') //Odchycení
chyby
             Console.log(err)}) //Výpis do konzole
}
```

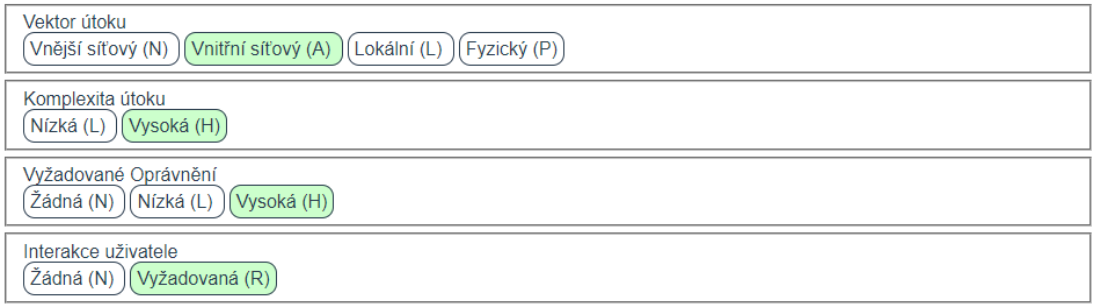
Pro zobrazení dat z backendu bude vytvořen nový komponent *DatabaseItem.vue*. V tomto komponentu bude vytvořena tabulka pro zobrazení hodnot, které jsou získány z rodičovského komponentu. Aby bylo možné předat data jednotlivých položek databáze, bude vytvořeno pomocí *v-for* několik instancí komponentu *DatabaseItem.vue*, kterému bude předána vždy jedna položka. Vrácená data jsou ve formátu JSON, v komponentu *DatabaseItem.vue* je tedy možné získat určité položky, dle názvů v modelu. Pro vyhledávání bude vytvořen textový vstup a tlačítko, které pomocí GET a určeného řetězce získá filtrovaná data z backendu. Pro vyhledáváni tedy bude vytvořena funkce, která bude zavolána po stisknutí tlačítka a změní obsah dříve vytvořeného pole. Dále je vhodné v této funkci vytvořit podmínku, která zavolá původní metodu pro získání dat, pokud textový vstup bude prázdný.

Vzhled

Jelikož všechny komponenty CVSS a OWASP Risk Rating Methodology budou sdílet stejný vzhled, je možné použít stejný styl pro všechny jejich komponenty. Vue.js rozlišuje přepínač *scoped*, díky tomu je možné daný styl omezit na daný komponent, nebo použít globálně pro celou aplikaci. Vzhled komponentů pro výpočet skóre bude tedy naimplementován v komponentu *App.vue*, hlavním komponentu aplikace. Jelikož je styl aplikován na celou aplikaci, ale je možné mít stejné tagy i v jiných komponentech s jiným stylem, bude přiřazen tagům v komponentech výpočtu skóre třída a případně ID. Díky tomu je možné mít více stejných tagů s jiným stylem, i když není použit přepínač *scoped*. Toho bude využito především u komponentu *Database.vue*.

Tag div uvnitř části *template* bude mít plnou hranici, se zaoblením rohů. Dále bude nastavena hodnotu výplně a okraje na malou hodnotu, například 5px a 2px. Tím bude docíleno toho, že ostatní elementy budou mít mezeru mezi hranicí a elementem, hranice půjde vidět i na stranách, a jednotlivé části skóre budou mít mezi sebou malou mezeru. Dále je třeba v komponentech *BaseScore.vue* a *EnviromentalScore.vue* rozdělit fieldsety na přibližně dvě poloviny, které budou vloženy do divů. Jelikož tyto dva komponenty mají velké množství faktorů, ale malé množství možných hodnot, je vhodné je tímto rozdělit na dva sloupce, aby tyto části nezabíraly zbytečně velký prostor. Aby byly obě poloviny stejné, bude nastavena šířka na 50 % a přepínač *float* na hodnotu *left*. Jelikož toto nastavení bude použito pro všechny tagy tohoto typu, bude nastaven styl tagu fieldset. Zde bude nastavena pozice textu na levou stranu a okraj na spodní straně, aby fieldesty měly mezi sebou mezery. Pro legendy fieldsetů bude nastaven spodní okraj na 5px. Pro tag label bude nastavena hranice na plnou čáru. Dále bude změněna výplň a okraj na 5 a 2 pixely. Jelikož všechny přepínače budou mít nastavenu třídu *radioButton*, je možné všechny nastavit takovým způsobem, aby nebyly vůbec zobrazeny. Toho je docíleno pomocí notace *display: none;* pro tuto třídu. Pro každý přepínač existuje štítek (label), který slouží k zobrazení názvu přepínače, a zároveň je možné na něj kliknout. Tyto štítky budou tedy použity jako tlačítka pro výběr. Štítkům bude nastavena plná čára

jako hranice a její zaoblení v rozích. Jelikož je vhodné, aby bylo možné rozeznat, která možnost je vybrána, bude štítek dále nastaven takovým způsobem, aby byla jeho barva a tvar kurzoru změněny v případě, že se nad ním nachází kurzor, či je vybrán. Bude tedy upraven styl *label:hover*, který určuje, že je upravován styl štítku v případě, že se nad ním nachází kurzor. Bude tedy změněna barva pozadí a typ kurzoru na *pointer*. Dále bude nastavena barvu pozadí štítku v případě, že bude jeho daný přepínač vybraný. Pro tento efekt bude použita notace *input:checked* + *label*, kde znaménko plus určuje, že bude vybrán první štítek, který následuje po tagu input. Notace *:checked* říká, že se jedná o přepínač, který je vybraný. Zde bude také nastavena barva pozadí. Následně bude nastaven styl tagu div, který obsahuje výstupy pro skóre a tlačítko pro výpočet tak, že bude zarovnán text na levou stranu. Pro všechna tlačítka bude nastavena velikost textu na 16px a pravý okraj na 2px pro případ, kdy by se nacházelo více tlačítek vedle sebe. Výsledný vzhled pro komponenty výpočtu skóre je na obr. 6.5.



Obrázek 6.5: Část vyplněného základního skóre

V souboru *Database.vue* bude nejprve upravován div, který obsahuje všechny části pro uložení, tedy vytvořenou tabulku a tlačítko pro uložení. Bude nastavena šířka na hodnotu, která umožňuje zobrazit celý CVSS vektor bez deformací. Dále budou nastaveny okraje jako automatické, což tento tag vycentruje, a jako poslední bude text zarovnán na levou stranu. U tabulky, kterou obsahuje, bude nastavena hranice a zarovnání textu na levou stranu. Dále bude nastaveno rozložení tabulky na automatické. Jako poslední bude nastaven spodní okraj na 3 pixely, aby se tabulka nepřekrývala s tlačítkem. U buněk tabulky bude nastavena spodní hranice, u buněk v posledním řádku však bude tato hranice odstraněna, aby na spodní straně tabulky nebyla dvojitá hranice. Dále pro buňky s názvy proměnných bude nastavena pravá hranici na plnou čáru a šířka na 20 %. Vzhled tabulky je na obr. 6.6.

Název	Označení zranitelnosti
Skóre CVSS	5.6
Vektor CVSS	AV:A/AC:H/PR:L/UI:R/S:U/C:H/I:L/A:L
Skóre OWASP	Střední
Datum	24.5.2021

Obrázek 6.6: Tabulka pro uložení

Pro část k zobrazení výstupu z databáze bude nastavena šířka, a podobně jako u předchozí části pro uložení dat, bude nastaven automatický okraj a zarovnaní textu vlevo. Nadpis této části bude vycentrován na střed a jako poslední bude nastavena velikost textu vstupu pro vyhledávání na 16px. V komponentu *DatabaseItem.vue* bude nastavena hranice, okraje na horní a spodní straně, zobrazení *inline-block* a šířka 100 %. Následně pro div s třídou *rowDiv* bude nastavena šířka také na 100 % a zobrazení *inline-flex*. Pro třídu *notLastRow* bude třeba nastavit šířku 25 % a spodní hranici na plnou čáru. Jednou z posledních úprav bude nastavení šířky divu třídy *vectorName* na 15 %. Každá položka z databáze se tedy zobrazí, viz obr. 6.7.

Název:	Test05	Datum:	11.4.2021
Skóre CVSS:	4.1	Skóre OWASP:	Žádná
Vektor CVSS:	AV:N/AC:H/PR:L/UI:R/S:C/C:L/I:L/A:L/E:U/RL:T/RC:R/CR:M/IR:M/AR:M/MAV:N/MAC:H/MPR:N/MUI:N/MS:U/MC:L/MI:N/MA:L		

Obrázek 6.7: Položka z databáze

7. TESTOVÁNÍ

Po implementaci může začít testování aplikace. Bude otestována jak funkčnost a správnost výpočtů, tak možnost ukládání a vyhledávání v databázi. Následně bude zjištěna základní bezpečnost aplikace samotné.

7.1 Testování funkčnosti

V této části bude otestována funkčnost a správnost výpočtů CVSS skóre, CVSS vektoru a OWASP skóre. Jelikož byla v předchozí kapitole vypočtena skóre pro několik zranitelností, budou tyto výsledky porovnány s výsledky vytvořené aplikace. U skóre dle OWASP Risk Rating Methodology je však třeba skóre znovu spočítat, jelikož nebyla použita část s faktory nositele hrozby. U těchto faktorů bude tedy vždy použito první hodnoty, a dále už hodnoty z předchozí kapitoly. Jelikož bylo v práci zpracováno pouze několik zranitelností, bude pro získání více dat pro testování použito některých bodů z OWASP ASVS.

ASVS je základ pro tvorbu a testování bezpečných webových aplikací [35]. Tento standard rozlišuje tři úrovně, kterých by měly aplikace dosáhnout, pro ideální bezpečnost aplikace [35]. Požadavky jsou strukturovány formou <kapitola>.<sekce>.<požadavek>.

Jelikož tyto body nejsou hodnoceny, je třeba pro ně určit hodnoty jednotlivých metrik skóre. Stejně jako u předchozích již hodnocených zranitelností bude vypočítáno pouze základní skóre. Pro test funkce budou použity některé požadavky z kapitol 2 a 3 tohoto standardu, které se zabývají bezpečností hesel, autentizací, jejím ověřením a managementem relací a jejich verifikací [36]. Příkladem z kapitoly 2 může být požadavek 2.1.7, který říká, že hesla mají být zkontrolována s databází prolomených hesel. Pokud tato podmínka není splněna, může se stát, že heslo bude jednoduše prolomitelné odkudkoliv, pokud má útočník přístup k prolomeným heslům. Tento požadavek lze velmi dobře zkombinovat např. s požadavkem 2.2.1, který se zabývá ochranou proti útokům hrubou silou. Pokud tato ochrana není implementována, může útočník tento útok použít pro prolomení hesla a získání důvěrných údajů uživatele. Podobným způsobem jsou ohodnoceny také ostatní požadavky druhé kapitoly.

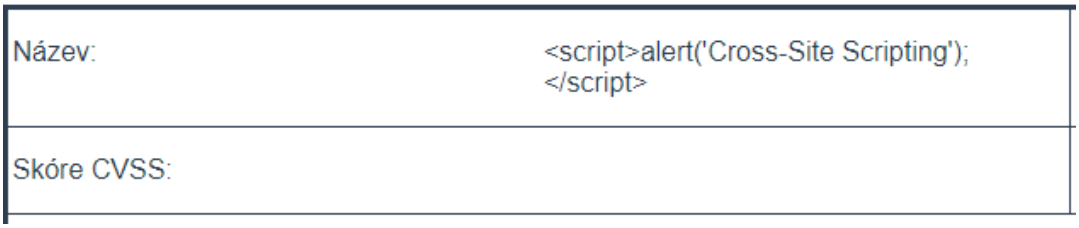
Hned prvním požadavkem třetí kapitoly je, aby aplikace nikdy neodhalila token relace v URL parametrech. Pokud tato podmínka není splněna, může útočník velmi lehce získat token uživatele a získat přístup k jeho relaci. Tím může jednoduše odhalit chráněná data. Podobnou zranitelností může být například CVE-2017-9280 a hodnocení je v podstatě stejné, jako hodnocení provedené implementovanou aplikací [37]. Podobným způsobem budou zhodnoceny i ostatní body této kapitoly. Dále pro všechny body bude proveden výpočet pomocí originálního kalkulátoru vytvořeného FIRST (Forum of Incident Response and Security Teams), pro porovnání hodnot.

Výsledky tohoto testování jsou v příloze A pro porovnání již vypočítaných zranitelností, a v příloze B pro některé body OWASP ASVS. Dále porovnávané výsledky a některé body ASVS budou uloženy do databáze, pro otestování její funkčnosti. Po uložení je třeba znovu načíst stránku, aby aplikace znovu získala data z databáze, jelikož se ukládaná data automaticky vkládají do množiny již získaných výsledů. Pokud se uložené zranitelnosti objeví ve výsledcích i po znovu načtení stránky, jsou data uložena v databázi. Dále pro testování vyhledávání v databázi, stačí pouze vyhledat určitý řetězec a porovnat, zda jsou zobrazeny všechny výsledky s tímto řetězcem.

7.2 Testování bezpečnosti

Jako první by bylo dobré se zmínit, že samotný framework Vue.js obsahuje obranu proti nejvíce používaným zranitelnostem. Nejdůležitější částí zde je, že framework upravuje vstup tak, aby nebylo možné využít většiny typů injekcí [38]. Toho dosahuje pomocí tzv. *escapingu*, to znamená, že se speciální znaky, např. <, převedou na řetězec několika jiných znaků, v případě < na <.

Pro testování budou použity některé zranitelnosti z OWASP Top Ten, které obsahuje nejčastěji přítomné zranitelnosti ve webových aplikacích. Je třeba vybrat zranitelnosti, které se mohou týkat vytvořené aplikace. Nejčastější zranitelností je injekce [39]. Jeden z nejznámějších druhů, tedy SQL injekce, již byla popsána v předchozích kapitolách. Proti této injekci je aplikace bezpečná z několika důvodů. Prvním z nich je, že samotný framework Django má proti některým zranitelnostem obranu [40]. Dalším důvodem je fakt, že s databází samotnou v podstatě není pracováno. Zatímco s databází samotnou pracuje framework, aplikace má k dispozici pouze výsledná získaná data. Další z velmi známých zranitelností je již dříve popsaný cross-site scripting. Jak již bylo popsáno v předchozí kapitole, útočník se snaží spustit kód JavaScriptu. Tuto zranitelnost je možné jednoduše otestovat v aplikaci, pokud je do pole s názvem zranitelnosti vepsán nějaký skript. Jelikož se název uložil, jak lze vidět na obr. 7.1, a v aplikaci se nezobrazilo žádné upozornění s napsanou zprávou, dá se předpokládat, že aplikace netrpí touto zranitelností.



Obrázek 7.1: Uložená položka se skriptem v názvu

Application Security Verification Standart

Jelikož aplikace nepracuje s žádnými citlivými daty, je pro ni dostačující úroveň 1. Na této úrovni aplikace nemusí splňovat velké množství podmínek tohoto standardu, např. kapitola 1, která se zabývá samotným designem a vývojem aplikace. Dále je také možné vypustit části, které nejsou pro aplikaci relevantní, jako např. tokeny uživatelů, ukládání finančních informací apod.

První důležitou kapitolou je kapitola 5. ASVS, která se zabývá ověřováním a sanitizací vstupů a výstupů. Kapitola 5.2 se zde zabývá tématem, které již bylo popsáno dříve, a to sanitizací dat [36]. Zde je důležité, aby vstupy aplikace byly ošetřeny takovým způsobem, aby nebylo možné jich zneužít. Jedná se tedy většinou o obranu proti různým druhům injekcí a spuštění uživatelského kódu. Tohoto efektu dosahuje obrana jak frameworku Vue.js, tak i frameworku Django [38], [40]. V této části je možné vynechat požadavek 5.2.3, který upravuje vstup před jeho předáním mailovému systému, jelikož žádný takovýto systém aplikace neobsahuje. Ze stejného důvodu budou vynechány některé body standardu.

Kapitola 7. ASVS se zaměřuje na zpracování chyb a logování. Zde je pro první úroveň potřeba splnit pouze tři požadavky. Požadavky 7.1.1 a 7.1.2 jsou v podstatě automaticky splněny [36]. Není použito žádné přihlašování ani žádné citlivé údaje a ani tyto údaje nejsou logovány či zpracovávány.

Bezpečnou komunikací se zabývá kapitola 9. ASVS. Pro první úroveň se jedná především o použití nejnovější verze TLS (Transport Layer Security) [36]. V tomto bodě má aplikace nedostatky, jelikož momentálně nepoužívá zabezpečenou komunikaci. Frontendová aplikace sice podporuje HTTPS, ale nemá platný certifikát. Backendová část momentálně není schopná tuto funkci použít, jelikož je stále používán defaultní server.

Kapitola 10.3 ASVS se zabývá integritou již funkčních aplikací a jejich aktualizace. Zde je důležitý bod 10.3.1, který se zabývá používáním nedůvěryhodných zdrojů a knihoven [36]. Aplikace používá pouze knihovny z důvěryhodných zdrojů, proto je tento požadavek splněn.

Další důležitou kapitolou je kapitola 13. ASVS, která se zabývá API [36]. Zde jsou splněny pouze dvě podmínky 13.1.3 a 13.2.1, které se zabývají výskytem citlivých informací v URL a omezení některých metod.

Kapitola 14. ASVS se zabývá samotnou konfigurací aplikace. První použitá sekce v této kapitole je sekce 14.2, kde aplikace nesplňuje hned první podmínku, aktuálnost komponentů [36]. Aplikace je napsána ve frameworku Vue.js verze 2, zatímco verze 3 už je momentálně dostupná. Další zajímavou částí je sekce 14.4, která se zabývá hlavičkami HTTP odpovědí. Zde je vhodné použít nástroj na testování API, například Postman. Po zadání adresy API, např. adresy, která vypíše všechny uložené zranitelnosti, je možné prohlédnout obsah odpovědi a její hlavičky, viz obr. 7.2. Z tohoto výpisu je možné získat veškeré informace pro sekci 14.4 a zjistit, které části jsou dle OWASP ASVS nedostačující.

KEY	VALUE
Date	Sun, 30 May 2021 13:52:37 GMT
Server	WSGIServer/0.2 CPython/3.8.0
Content-Type	application/json
Content-Disposition	attachment; filename="api.json"
Vary	Accept, Origin, Cookie
Allow	OPTIONS, GET
X-Frame-Options	DENY
Content-Length	2174
Content-Security-Policy	default-src 'none'; script-src 'self';
X-Content-Type-Options	nosniff
Referrer-Policy	same-origin

Obrázek 7.2: Výpis hlaviček z aplikace Postman

Z výsledku testování bezpečnosti vytvořené aplikace vyplynulo, že aplikace nesplňuje veškeré požadavky pro dosažení standardu první úrovně dle ASVS. Tento standard není zcela splněn z důvodu absence částí, které jsou nutné pro hodnocení. Vytvořená aplikace však nevyžaduje implementaci těchto částí, protože aplikace nepoužívá a nezpracovává citlivá data. Tento standard slouží spíše jako doporučení vývojářům, není tedy nutné veškeré body striktně dodržovat.

8. ZÁVĚR

V práci byla provedena analýza dvou metodik hodnocení zranitelností a jejich faktorů. Byly také určeny jejich výhody a nevýhody. Dále byly tyto metodiky porovnány s další, méně známou metodikou a bylo popsáno, k jakému účelu jsou vhodné.

Byly popsány známe zranitelnosti webových aplikací. Pro každou zranitelnost byly uvedeny možné metody obrany a bylo provedeno jejich hodnocení pomocí dvou vybraných metod, a to CVSS a OWASP Risk Rating Methodology.

Nejdůležitější částí byl návrh a implementace webové aplikace, obsahující dvě metodiky hodnocení zranitelnosti. Byl proveden návrh aplikace ve frameworku Vue.js. Aplikace používá modul vue router pro pohyb mezi hlavními komponenty. Hlavní komponenty jsou Cvss.vue, Owasp.vue a Database.vue. Tyto hlavní komponenty obsahují další komponenty, ve kterých jsou implementovány výpočty jednotlivých částí skóre. Komponent Databse.vue obsahuje vstup pro vyhledávání ukládání do databáze a komponent pro zobrazení jednotlivých uložených zranitelností. Byl také proveden návrh vzhledu a backendu aplikace. Backend se skládá z databáze PostgreSQL a API, implementovaného pomocí frameworku Django. Byl také vytvořen model pro API, který obsahuje název, dva typy skóre, CVSS vektor a datum uložení hodnocení. Dále bylaprovedena implementace samotné aplikace. Byl implementován backend, napsaný ve frameworku Django, a vytvořena databáze PostgreSQL. Pro API byl vytvořen model a serializátor modelu. Jednou z velmi důležitých částí byla implementace metod API (*views*). V implementaci frontendu byly vytvořeny jednotlivé komponenty a příslušné metody pro výpočty skóre. Bylo implementováno předávání dat, a nakonec metody pro volání API v komponentu Database.vue.

V následující kapitole byla tato aplikace otestována. Nejprve byla otestována samotná funkčnost aplikace, kde byly vypočteny skóre zranitelností, který byly ohodnoceny v předchozích kapitolách. Tyto hodnoty byly porovnány se správnými hodnotami, pro určení, zda aplikace pracuje správně. Byla také otestována funkčnost API a ukládání do databáze. Následovalo testování bezpečnosti aplikace pomocí některých známých zranitelností, a také pomocí OWASP ASVS.

Další kroky, které by bylo možné provést jsou vyhledání více metod hodnocení zranitelností a implementace těchto metod do aplikace, nebo tvorba produkčního buildu aplikace a její deployment na server, společně s vytvořením platných certifikátů, a tím zvýšení zabezpečení aplikace a API.

Výsledkem práce je vytvořená webová aplikace implementující hodnocení závažnosti zranitelnosti pomocí metodologie CVSS a OWASP Risk Rating Methodology. Vypočtené hodnoty je také možné uložit do interní databáze aplikace. Samotná aplikace byla otestována pomocí metodologie OWASP ASVS.

LITERATURA

[1] 20,000+ new vulnerability reports predicted for 2020, shattering previous records. Help Net Security [online]. Kastav: Help Net Security, 2020 [cit. 2020-12-06]. Dostupné z: https://www.helpnetsecurity.com/2020/07/22/vulnerability-reports-2020/

[2] Common Vulnerability Scoring System SIG. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-11]. Dostupné z: https://www.first.org/cvss/

[3] PECL, David. Návrh metody pro hodnocení bezpečnostních zranitelností systémů. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/125988. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Tomáš Gerlich.

[4] Common Vulnerability Scoring System v3.1: Specification Document. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-11]. Dostupné z: https://www.first.org/cvss/v3.1/specification-document

[5] Common Vulnerability Scoring System v1 Archive. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-18]. Dostupné z: https://www.first.org/cvss/v1/

[6] CVSS-SIG Version 2 History. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-18]. Dostupné z: https://www.first.org/cvss/v2/history

[7] CVSS v2 Complete Documentation. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-11]. Dostupné z: https://www.first.org/cvss/v2/guide

[8] CVSS v3.0 Specification Document. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-11]. Dostupné z: https://www.first.org/cvss/v3.0/specification-document

[9] CVSS v3.1 Calculator Use & Design. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2020 [cit. 2020-10-11]. Dostupné z: https://www.first.org/cvss/v3.1/use-design#Changelog

[10] M., Art, Dave D., Dale R., et al. List of Potential Improvements for CVSS v4.0. CVSS v4.0 Work Items [online]. Forum of Incident Response and Security Teams, c2015—2021 [cit. 2021-5-24]. Dostupné z: https://docs.google.com/document/d/1qmmk9TQulW9d1cuipu_ziXDX0pUswbZ1WSQyynHbvKU/edit#

[11] ABOUD, Jeff. Why You Need to Stop Using CVSS for Vulnerability Prioritization. TENABLE BLOG [online]. Columbia: Tenable, 2020 [cit. 2021-5-24]. Dostupné z: https://www.tenable.com/blog/why-you-need-to-stop-using-cvss-for-vulnerability-prioritization
[12] OWASP Risk Rating Methodology. OWASP Foundation [online]. c2020 [cit. 2020-10-18]. Dostupné z: https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
[13] Bugcrowd's Vulnerability Rating Taxonomy. Bugcrowd [online]. Bugcrowd, c2020 [cit. 2020-11-28]. Dostupné z: https://bugcrowd.com/vulnerability-rating-taxonomy#usage-guide
[14] VULNERABILITY PRIORITIZATION AT BUGCROWD. Bugcrowd [online]. Bugcrowd, 2015 [cit. 2020-11-28]. Dostupné z: https://www.bugcrowd.com/blog/vulnerability-prioritization-at-bugcrowd/
[15] SQL Injection. OWASP Foundation [online]. OWASP Foundation, kingthorin, c2020 [cit. 2020-10-30]. Dostupné z: https://owasp.org/www-community/attacks/SQL_Injection
[16] SQL Injection Prevention Cheat Sheet. OWASP Cheat Sheet Series [online]. CheatSheets Series Team, c2020 [cit. 2020-10-30]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
[17] Common Vulnerability Scoring System v3.1: Examples. FIRST - Improving Security Together [online]. Forum of Incident Response and Security Teams, c2015—2021 [cit. 2021-5-30]. Dostupné z: https://www.first.org/cvss/examples
[18] Cross Site Scripting (XSS). OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-10-31]. Dostupné z: https://owasp.org/www-community/attacks/xss/
[19] A7:2017-Cross-Site Scripting (XSS). OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-10-31]. Dostupné z: https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)
[20] DOM Based XSS. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-10-31]. Dostupné z: https://owasp.org/www-community/attacks/DOM_Based_XSS
[21] Cross Site Scripting Prevention Cheat Sheet. OWASP Cheat Sheet Series [online]. CheatSheets Series Team, c2020 [cit. 2020-10-31]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
[22] KIRSTENS. Cross Site Request Forgery (CSRF). OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-01]. Dostupné z: https://owasp.org/www-community/attacks/csrf

[23] Cross-Site Request Forgery Prevention Cheat Sheet. OWASP Cheat Sheet Series [online]. CheatSheets Series Team, c2020 [cit. 2020-11-01]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[24] CVE-2019-1958 Detail. NATIONAL VULNERABILITY DATABASE [online]. Gaithersburg: National Institute of Standarts and Technology, 2019 [cit. 2020-12-06]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2019-1958
[25] CRLF Injection. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-15]. Dostupné z: https://owasp.org/www-community/vulnerabilities/CRLF_Injection
[26] HTTP Response Splitting. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-15]. Dostupné z: https://owasp.org/www-community/attacks/HTTP_Response_Splitting
[27] Log Injection. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-15]. Dostupné z: https://owasp.org/www-community/attacks/Log_Injection
[28] CVE-2019-19330 Detail. NATIONAL VULNERABILITY DATABASE [online]. Gaithersburg: National Institute of Standarts and Technology, 2020 [cit. 2020-11-15]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2019-19330
[29] Path Traversal. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-22]. Dostupné z: https://owasp.org/www-community/attacks/Path_Traversal
[30] Denial of Service. OWASP Foundation [online]. OWASP Foundation, c2020 [cit. 2020-11-15]. Dostupné z: https://owasp.org/www-community/attacks/Denial_of_Service
[31] CVE-2018-10531 Detail. NATIONAL VULNERABILITY DATABASE [online]. Gaithersburg: National Institute of Standarts and Technology, 2021 [cit. 2021-5-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2018-10531
[32] Introduction. Vue.js [online]. Evan You, c2014—2020 [cit. 2020-11-27]. Dostupné z: https://vuejs.org/v2/guide/
[33] ENCODE. Django REST framework. Django REST framework [online]. Encode, c2011-2021 [cit. 2021-5-24]. Dostupné z: https://www.django-rest-framework.org/
[34] ZABRISKIE, Matt, Nick URALTSEV, Emily MOREHOUSE, et al. Axios. Axios/axios: Promise based HTTP client for the browser and node.js [online]. c2014-2021 [cit. 2021-5-30]. Dostupné z: https://github.com/axios/axios

[35] OWASP Application Security Verification Standard. OWASP Application Security Verification Standard [online]. OWASP Foundation, c2021 [cit. 2021-04-22]. Dostupné z: https://owasp.org/www-project-application-security-verification-standard/
[36] OWASP Application Security Verification Standard 4.0.2. OWASP Foundation [online]. OWASP Foundation, 2020 [cit. 2021-5-2]. Dostupné z: https://github.com/OWASP/ASVS/raw/v4.0.2/4.0/OWASP%20Application%20Security%20Verification%20Standard%204.0.2-en.pdf
[37] CVE-2017-9280 Detail. NATIONAL VULNERABILITY DATABASE [online]. Gaithersburg: National Institute of Standarts and Technology, 2019 [cit. 2021-4-30]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-9280
[38] Security — Vue.js. Security — Vue.js [online]. Evan You, c2014-2021 [cit. 2021-04-22]. Dostupné z: https://vuejs.org/v2/guide/security.html#What-Vue-Does-to-Protect-You
[39] OWASP Top Ten. OWASP Top Ten [online]. OWASP Foundation, c2021 [cit. 2021-04-22]. Dostupné z: https://owasp.org/www-project-top-ten/
[40] Security in Django. Security in Django [online]. Django Software Foundation, c2005-2021 [cit. 2021-04-22]. Dostupné z: https://docs.djangoproject.com/en/3.2/topics/security/

SEZNAM SYMBOLŮ A ZKRATEK

Zkratky:

CVE	Common Vulnerabilities and Exposures
NVD	National Vulnerability Database
OWASP	Open Web Application Security Project
API	Application Programming Interface
ASVS	Application Security Verification Standard
CVSS	Common Vulnerability Scoring Systém
AV	Attack Vector
L	Local
OS	Operační Systém
VRT	Vulnerability Rating Taxonomy
CWE	Common Weakness Enumeration
SQL	Structured Query Language
XSS	Cross-site Scripting
HTML	Hypertext Markup Language
DOM	Document Object Model
URL	Uniform Resource Locator
CRLF	Carriage Return Line Feed
HTTP	Hypertext Transfer Protocol
DoS	Denial of Sevice
CSS	Cascading Sytle Sheets
REST	Representational State Transfer
IDE	Integrated Development Enviroment
CORS	Cross-Origin Resource Sharing
CSP	Content Security Policy
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
XML	Extensible Markup Language
FIRST	Forum of Incident Response and Security Teams
TLS	Transport Layer Security

Symboly:

E	Exploitovatelnost
VU	Vektor útoku
KU	Komplexita útoku
PO	Požadované oprávnění
IU	Interakce uživatele
DSS	Sub skóre dopadu
DD	Dopad na důvěrnost
DI	Dopad na integritu
DO	Dopad na dostupnost
D	Dopad
ZS	Základní skóre
Min	Minimum
RndU	Roundup
DS	Dočasné skóre
KE	Kvalita exploitu
UN	Úroveň nápravy
DR	Důvěrnost reportu
MDSS	Sub skóre modifikovaného dopadu
PDD	Požadavek na důvěrnost
MDD	Modifikovaný dopad na důvěrnost
PDI	Požadavek na integritu
MDI	Modifikovaný dopad na integritu
PDO	Požadavek na dostupnost
MDO	Modifikovaný dopad na dostupnost
MD	Modifikovaný dopad
ME	Modifikovaná exploitovatelnost
MVU	Modifikovaný vektor útoku
MKU	Modifikovaná komplexita útoku
MPO	Modifikované požadované oprávnění
MIU	Modifikovaná interakce uživatele
SP	Skóre prostředí

SEZNAM PŘÍLOH

Příloha A - Zranitelnosti ohodnocené aplikací

A.1 Tabulka hodnot vypočtených aplikací

Název	CVSS Skóre	OWASP RRM Skóre	CVSS Vektor
SQL Injection	10,0	Vysoká	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
XSS	6,1	Střední	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CSRF	8,8	Vysoká	AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
CRLF Injection	9,8	Vysoká	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Path Traversal	7,5	Střední	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
DoS	7,5	Střední	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

A.2 Tabulka správných hodnot

Název	CVSS Skóre	OWASP RRM Skóre	CVSS Vektor
SQL Injection	10,0	Vysoká	AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H
XSS	6,1	Střední	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CSRF	8,8	Vysoká	AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
CRLF Injection	9,8	Vysoká	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Path Traversal	7,5	Střední	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
DoS	7,5	Střední	AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

Příloha B - Hodnocení bodů OWASP ASVS

B.1 Tabulka hodnot vypočtených aplikací

Požadavek	CVSS Skóre	CVSS Vektor	Souhlasí se správným výpočtem
2.1.6	7,8	AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	Ano
2.1.7	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
2.2.1	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
2.2.3	8,1	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:H	Ano
2.4.1	5,9	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
2.5.1	9,1	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N	Ano
2.5.2	7,4	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N	Ano
2.5.3	9,8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
2.5.4	9,8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
2.5.5	8,8	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	Ano
2.10.1	8,8	AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
2.10.2	8,8	AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
3.1.1	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
3.2.1	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
3.2.3	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
3.2.4	7,5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N	Ano
3.3.1	4,4	AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N	Ano
3.3.2	4,4	AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N	Ano
3.3.3	8,3	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:H	Ano
3.3.4	8,3	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:L/A:H	Ano
3.4.1	3,7	AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N	Ano
3.4.2	5,3	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N	Ano
3.4.3	3,1	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N	Ano
3.4.4	6,5	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:L/A:N	Ano
3.4.5	7,7	AV:N/AC:H/PR:L/UI:N/S:C/C:H/I:L/A:L	Ano
3.5.1	7,7	AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:N/A:N	Ano
3.5.2	9,8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
3.5.3	8,1	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H	Ano
3.6.1	5,5	AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N	Ano
3.7.1	8,8	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H	Ano